**Publicis Sapient**

Week 1 Assignment

Name Prashant Thakur

Github profile

<https://github.com/PrashantThakurNitP>

Repository used:

https://github.com/PrashantThakurNitP/prashant\_sapient\_assignment

**Q1. Git-Sub:**

-------------------------------------------------------------------------------------------------------------------------------------

**a. Basic usage using the CLI**

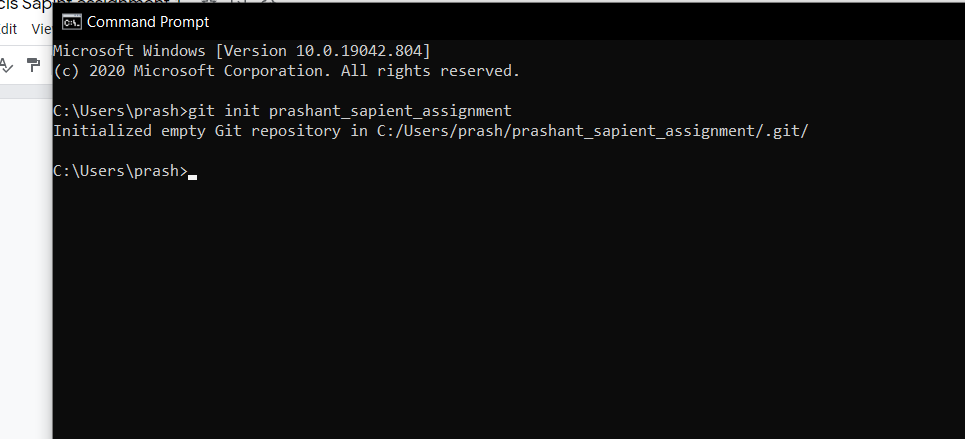
---------------------------------------------------------------------------------------------------------------------------

**i. Setup a Loco I Repository**

Answer :

C:\Users\prash>git init prashant\_sapient\_assignment

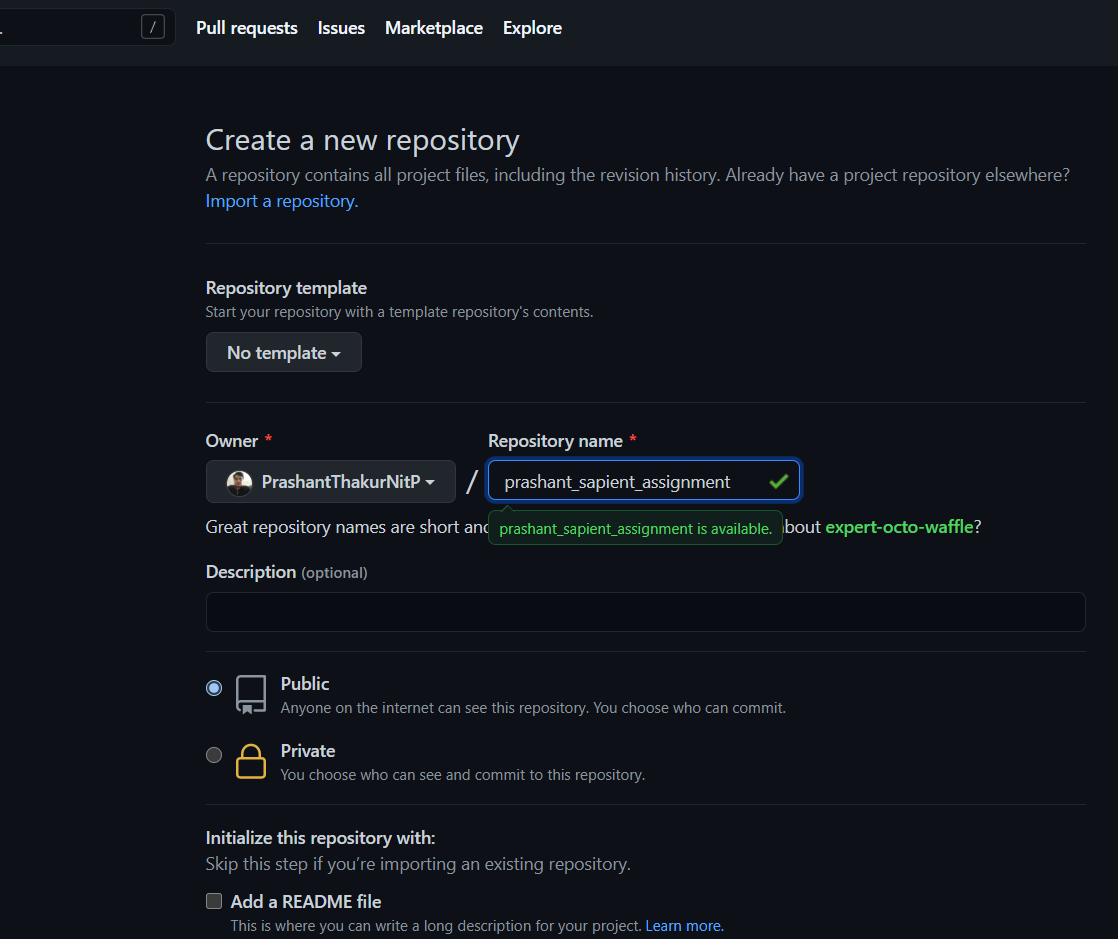
Initialized empty Git repository in C:/Users/prash/prashant\_sapient\_assignment/.git/

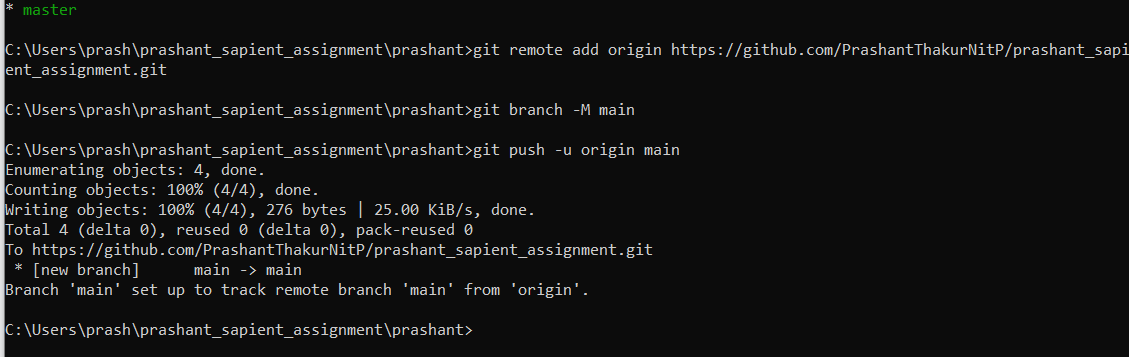


C:\Users\prash>

-------------------------------------------------------------------------------------------------------------------------------

**ii. Setup a Remote Repository**





C:\Users\prash\prashant\_sapient\_assignment\prashant>git remote add origin https://github.com/PrashantThakurNitP/prashant\_sapient\_assignment.git

C:\Users\prash\prashant\_sapient\_assignment\prashant>git branch -M main

C:\Users\prash\prashant\_sapient\_assignment\prashant>git push -u origin main

Enumerating objects: 4, done.

Counting objects: 100% (4/4), done.

Writing objects: 100% (4/4), 276 bytes | 25.00 KiB/s, done.

Total 4 (delta 0), reused 0 (delta 0), pack-reused 0

To https://github.com/PrashantThakurNitP/prashant\_sapient\_assignment.git

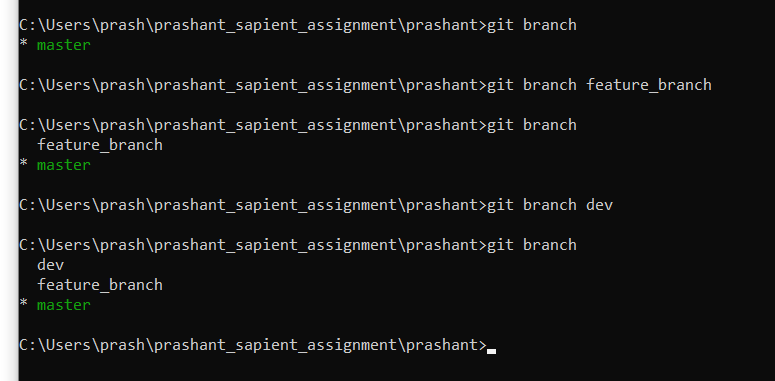
\* [new branch] main -> main

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

-------------------------------------------------------------------------------------------------------------------------------

**iii. Create Local branches (Feature branch, Dev. branch, QA Branch, Master / Procl. Branch, Delivery Branches)**

Answer



C:\Users\prash\prashant\_sapient\_assignment\prashant>git branch feature\_branch

C:\Users\prash\prashant\_sapient\_assignment\prashant>git branch

feature\_branch

\* master

C:\Users\prash\prashant\_sapient\_assignment\prashant>git branch dev

C:\Users\prash\prashant\_sapient\_assignment\prashant>git branch

dev

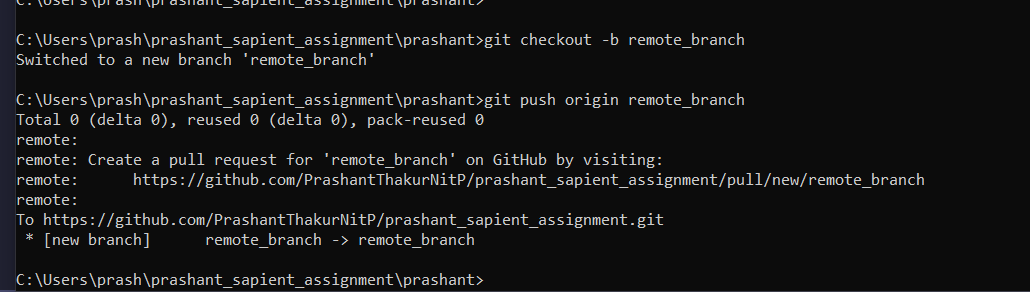
feature\_branch

\* master

C:\Users\prash\prashant\_sapient\_assignment\prashant>

-------------------------------------------------------------------------------------------------------------------

**iv. Create Remote branches**



C:\Users\prash\prashant\_sapient\_assignment\prashant>git checkout -b remote\_branch

Switched to a new branch 'remote\_branch'

C:\Users\prash\prashant\_sapient\_assignment\prashant>git push origin remote\_branch

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

remote:

remote: Create a pull request for 'remote\_branch' on GitHub by visiting:

remote: https://github.com/PrashantThakurNitP/prashant\_sapient\_assignment/pull/new/remote\_branch

remote:

To https://github.com/PrashantThakurNitP/prashant\_sapient\_assignment.git

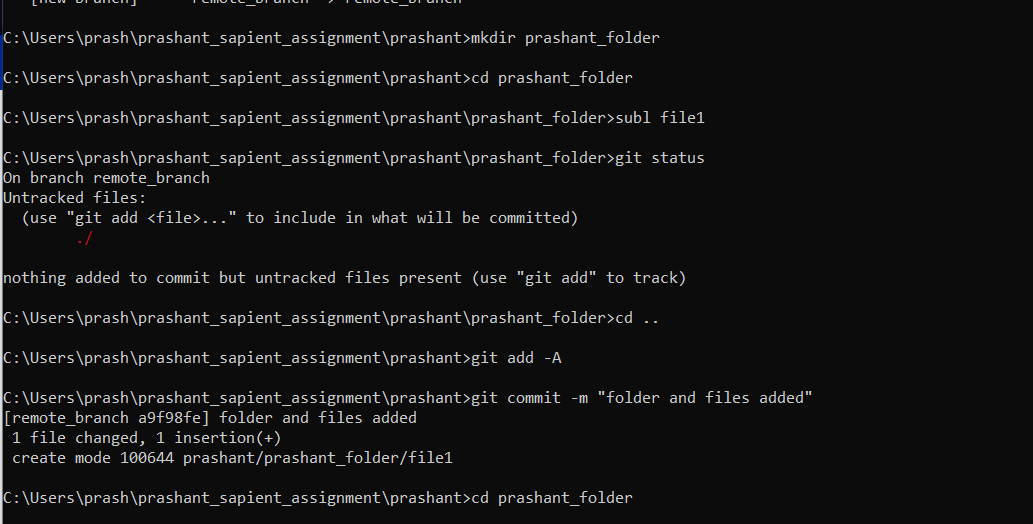
\* [new branch] remote\_branch -> remote\_branch

C:\Users\prash\prashant\_sapient\_assignment\prashant>

-----------------------------------------------------------------------------------------------------------------

**v) Acld files, Make changes to existing files, Acld folders, Remove folders, remove lifes**

**Ans**



C:\Users\prash\prashant\_sapient\_assignment\prashant>mkdir prashant\_folder

C:\Users\prash\prashant\_sapient\_assignment\prashant>cd prashant\_folder

C:\Users\prash\prashant\_sapient\_assignment\prashant\prashant\_folder>subl file1

C:\Users\prash\prashant\_sapient\_assignment\prashant\prashant\_folder>git status

On branch remote\_branch

Untracked files:

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

./

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

C:\Users\prash\prashant\_sapient\_assignment\prashant\prashant\_folder>cd ..

C:\Users\prash\prashant\_sapient\_assignment\prashant>git add -A

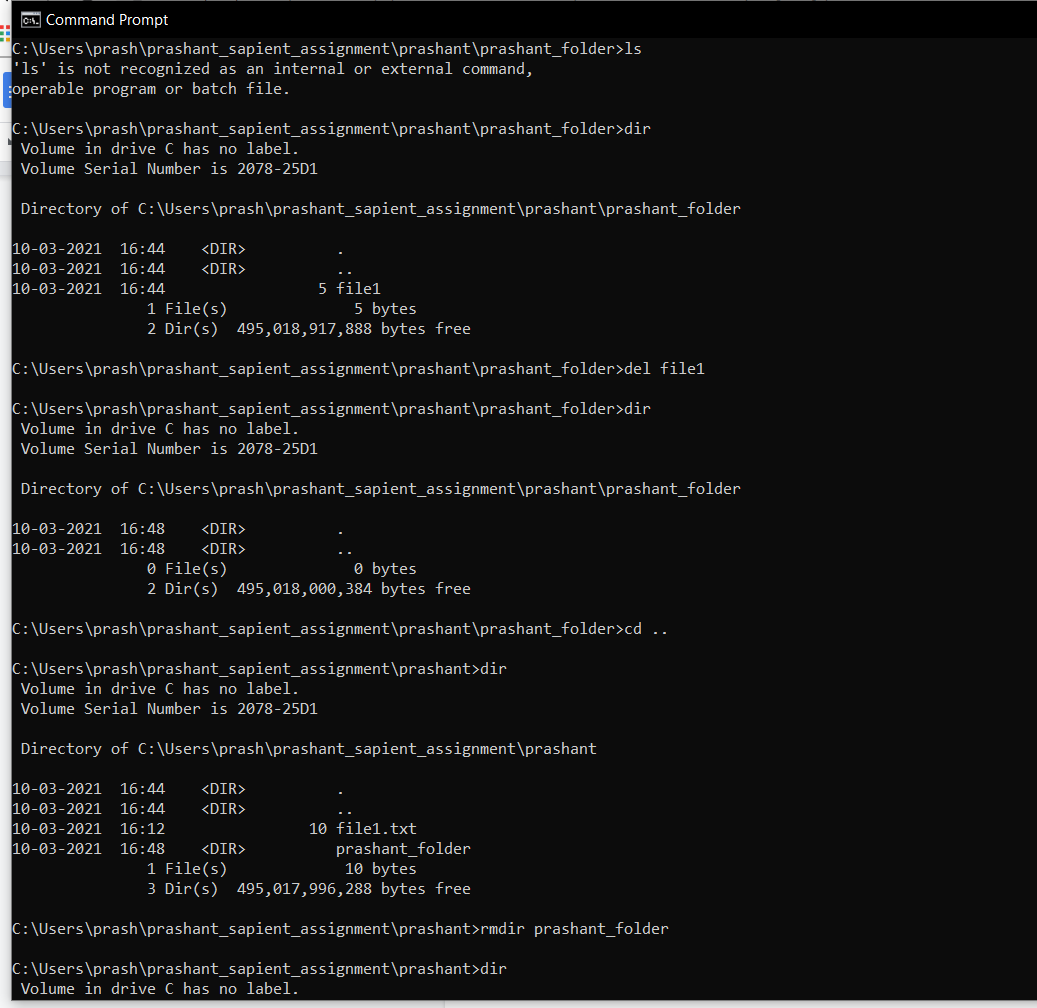
C:\Users\prash\prashant\_sapient\_assignment\prashant>git commit -m "folder and files added"

[remote\_branch a9f98fe] folder and files added

1 file changed, 1 insertion(+)

create mode 100644 prashant/prashant\_folder/file1

C:\Users\prash\prashant\_sapient\_assignment\prashant>cd prashant\_folder

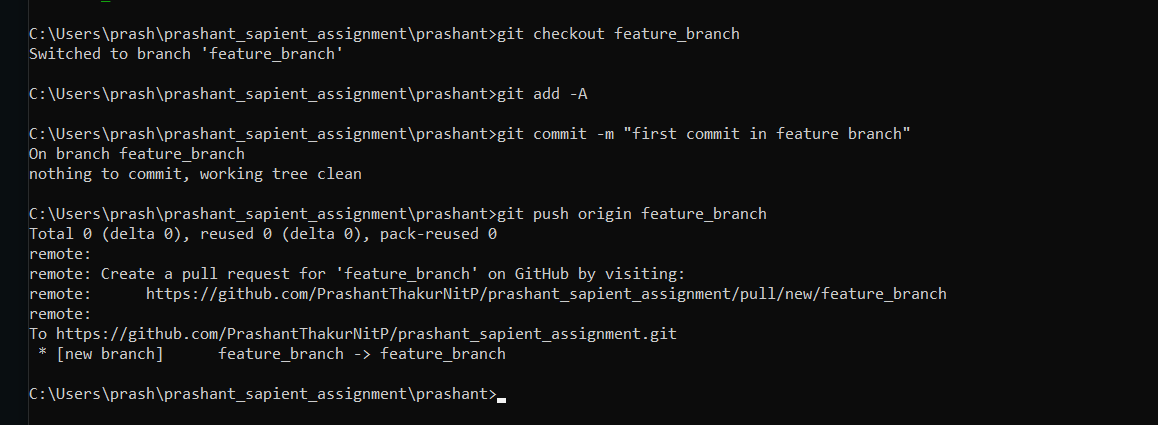


C:\Users\prash\prashant\_sapient\_assignment\prashant\prashant\_folder>del file1

C:\Users\prash\prashant\_sapient\_assignment\prashant>rmdir prashant\_folder

------------------------------------------------------------------------------------------------------------------------

**Qvi )Check-in, Stage, Commit, Push files into Feature Branch**



C:\Users\prash\prashant\_sapient\_assignment\prashant>git checkout feature\_branch

Switched to branch 'feature\_branch'

C:\Users\prash\prashant\_sapient\_assignment\prashant>git add -A

C:\Users\prash\prashant\_sapient\_assignment\prashant>git commit -m "first commit in feature branch"

On branch feature\_branch

nothing to commit, working tree clean

C:\Users\prash\prashant\_sapient\_assignment\prashant>git push origin feature\_branch

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

remote:

remote: Create a pull request for 'feature\_branch' on GitHub by visiting:

remote: https://github.com/PrashantThakurNitP/prashant\_sapient\_assignment/pull/new/feature\_branch

remote:

To https://github.com/PrashantThakurNitP/prashant\_sapient\_assignment.git

\* [new branch] feature\_branch -> feature\_branch

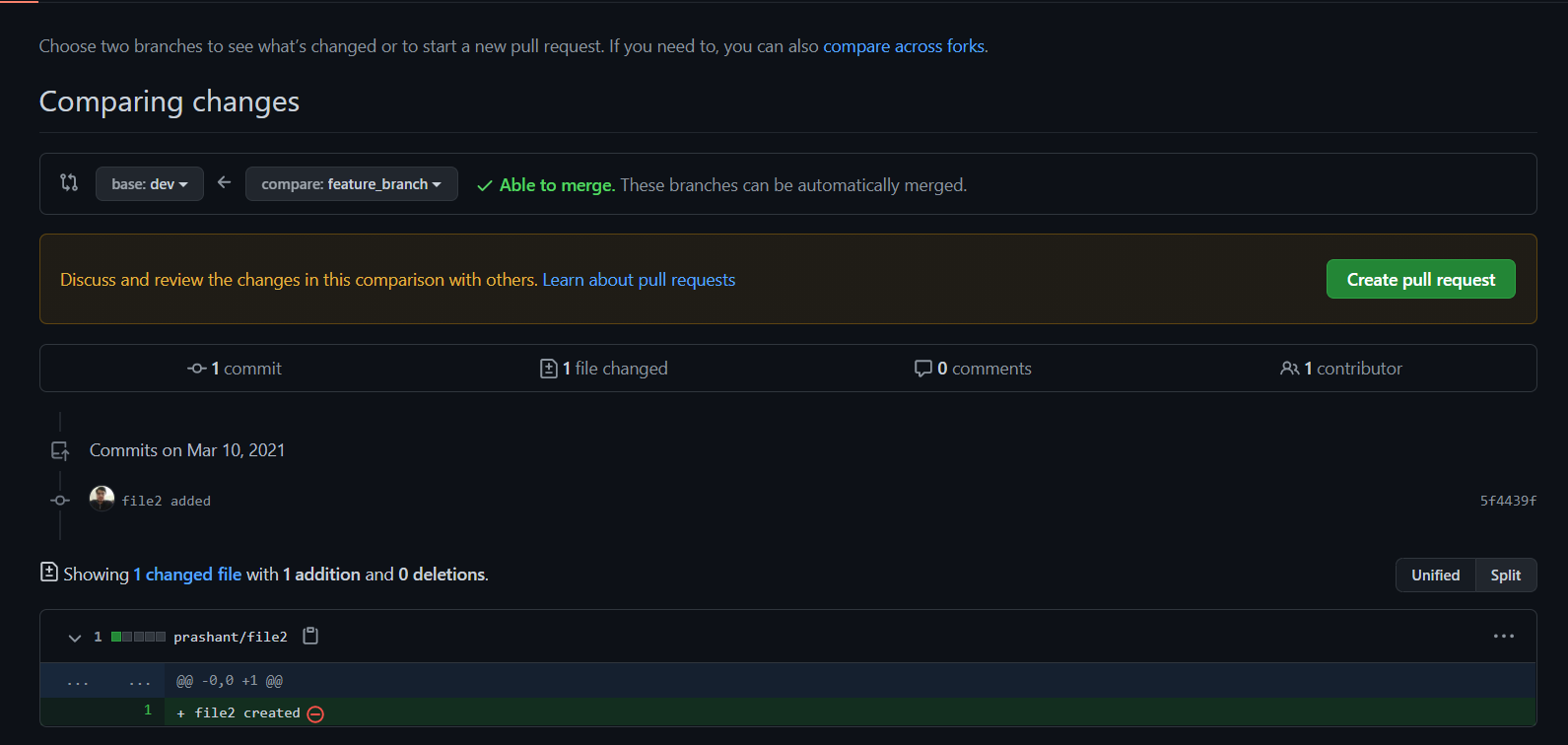
-------------------------------------------------------------------------------------------------------------------------------

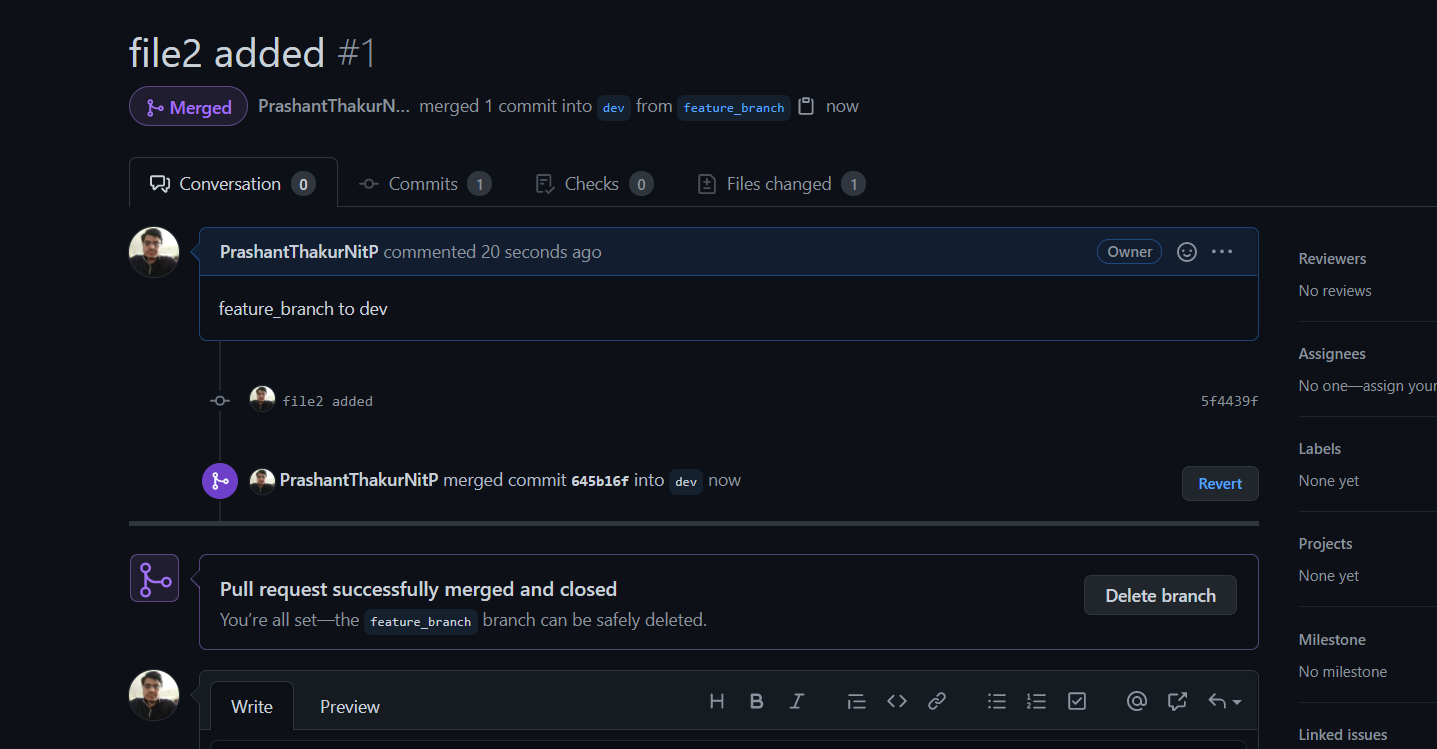
**vii. Promote code from Feature branch to Dev. branch via Pull requests**

Pull request is way to move code from one branch to another branch

It is advised to work on different feature branch and create pull request to merge code into development branch.

From development branch working code can shifted to production branch





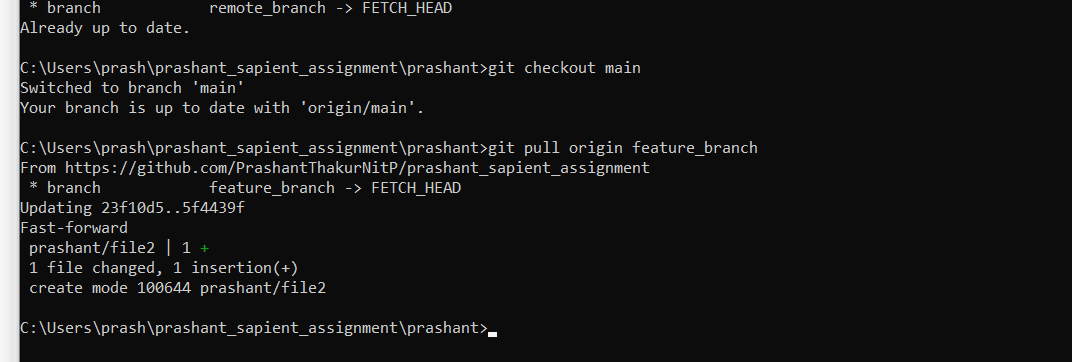
-----------------------------------------------------------------------------------------------------------------------------

**viii)Check-out the latest code from remote branch to local branch**

Answer

git pull origin feature\_branch

We can fetch code from remote repository by pull option and specifying remote repository name from which to pull



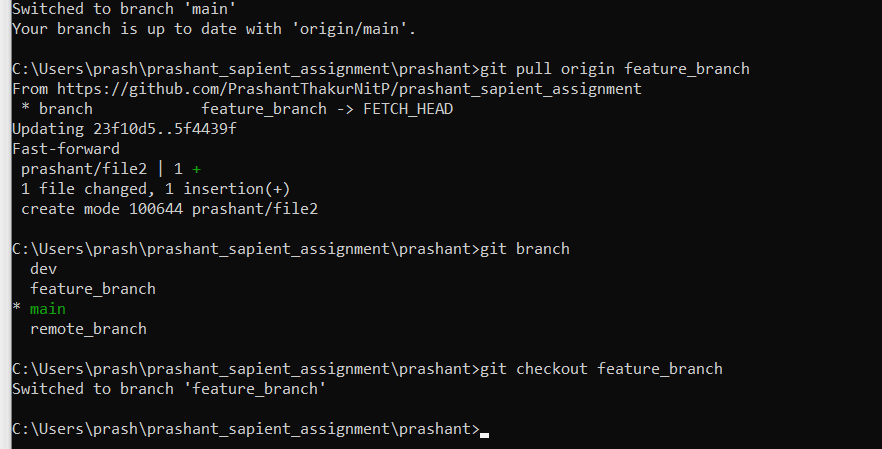
-------------------------------------------------------------------------------------------------------------------------------

**Q ix Explore the difference. between Checkout vs Pull**

Ans :

pull contacts the remote repository identified by origin and looks for updates. It fetches any updates and then merges the changes into the target branch. It does not create a new branch.

The git checkout command lets you navigate between the branches created by git branch. Checking out a branch updates the files in the working directory to match the version stored in that branch, and it tells Git to record all new commits on that branch. Think of it as a way to select which line of development you’re working on.



-------------------------------------------------------------------------------------------------------------------------------

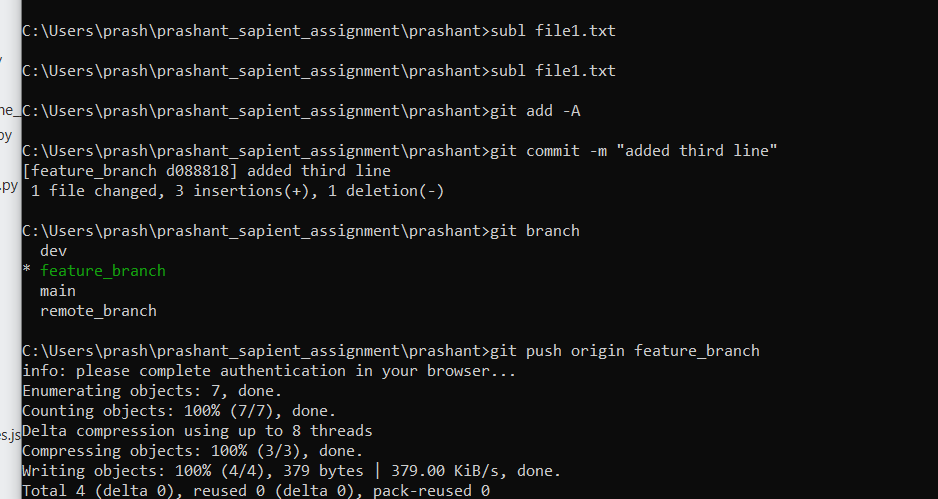
**Q x)Get two people to make changes to the same file, check-in & handle merge conflicts**

Ans

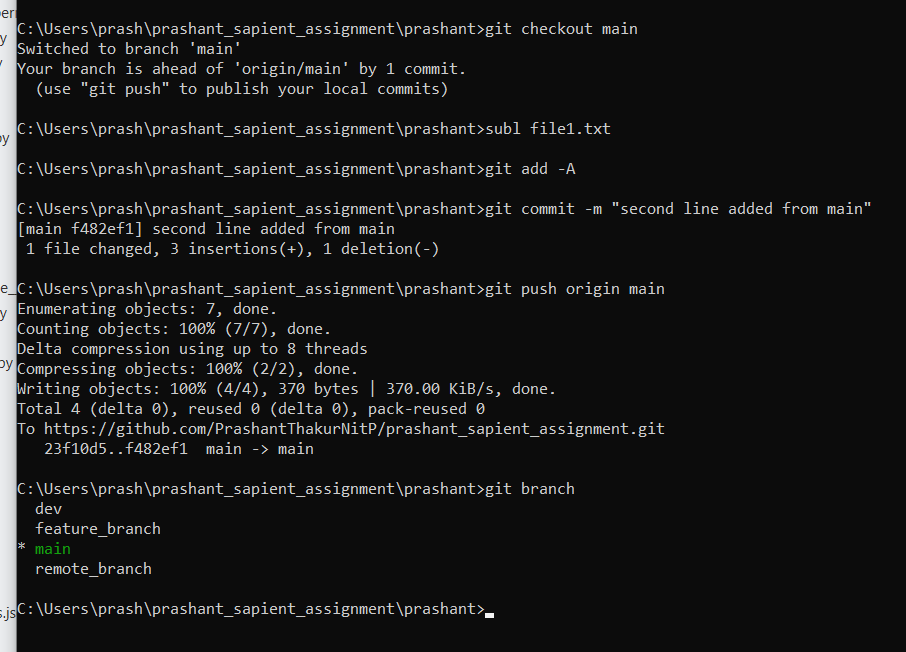
When two people work on same file at same time merger conflict may arise.

Hence it is advised to work on separate branches and merge them if need arise

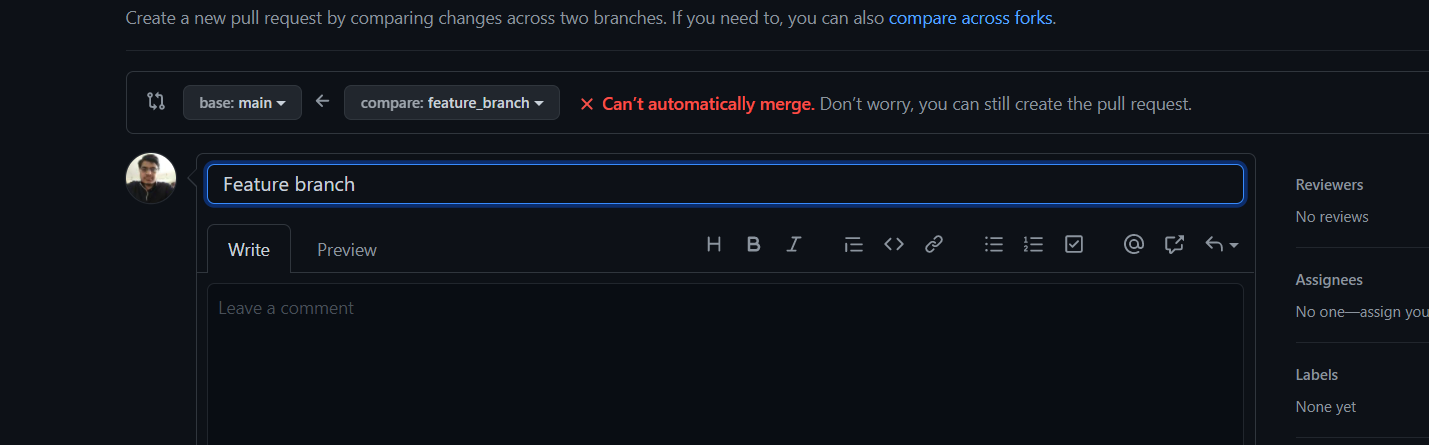
Made some changes in file1.txt from feature branch and pushed it



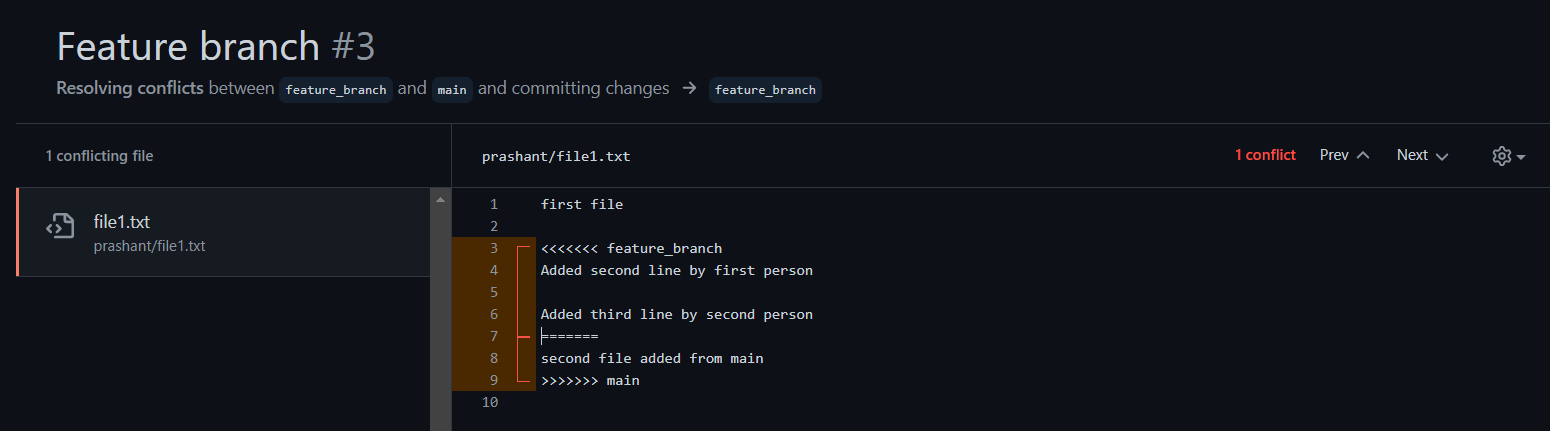
Made some changes in main branch in same file and pushed it



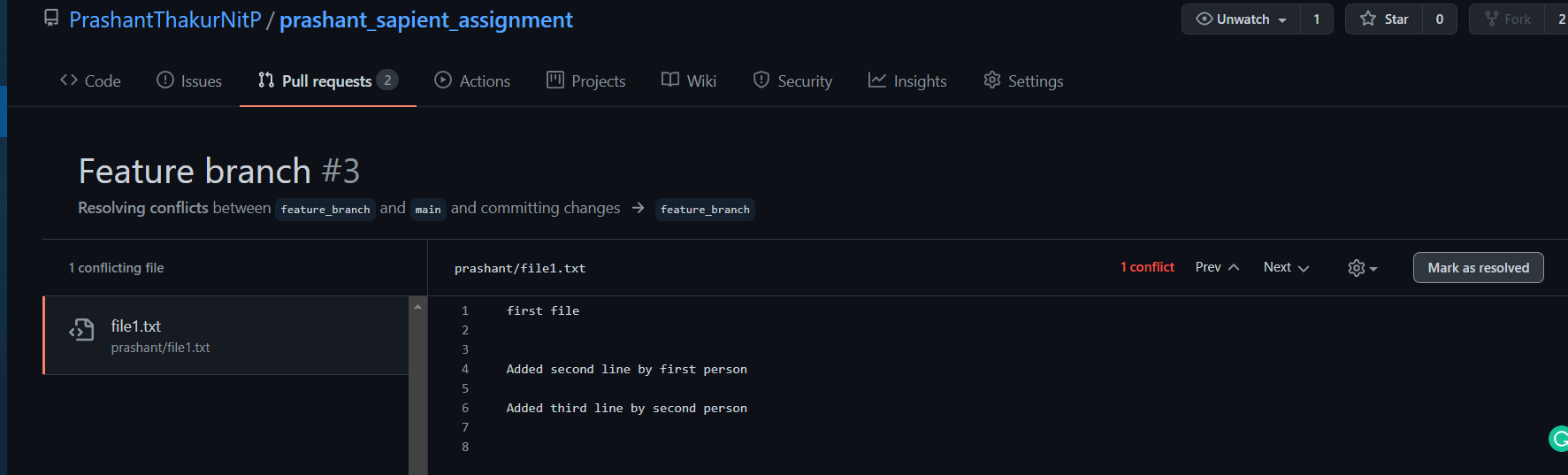
Now generator pull request for merging



Conflict occur in file file1.txt

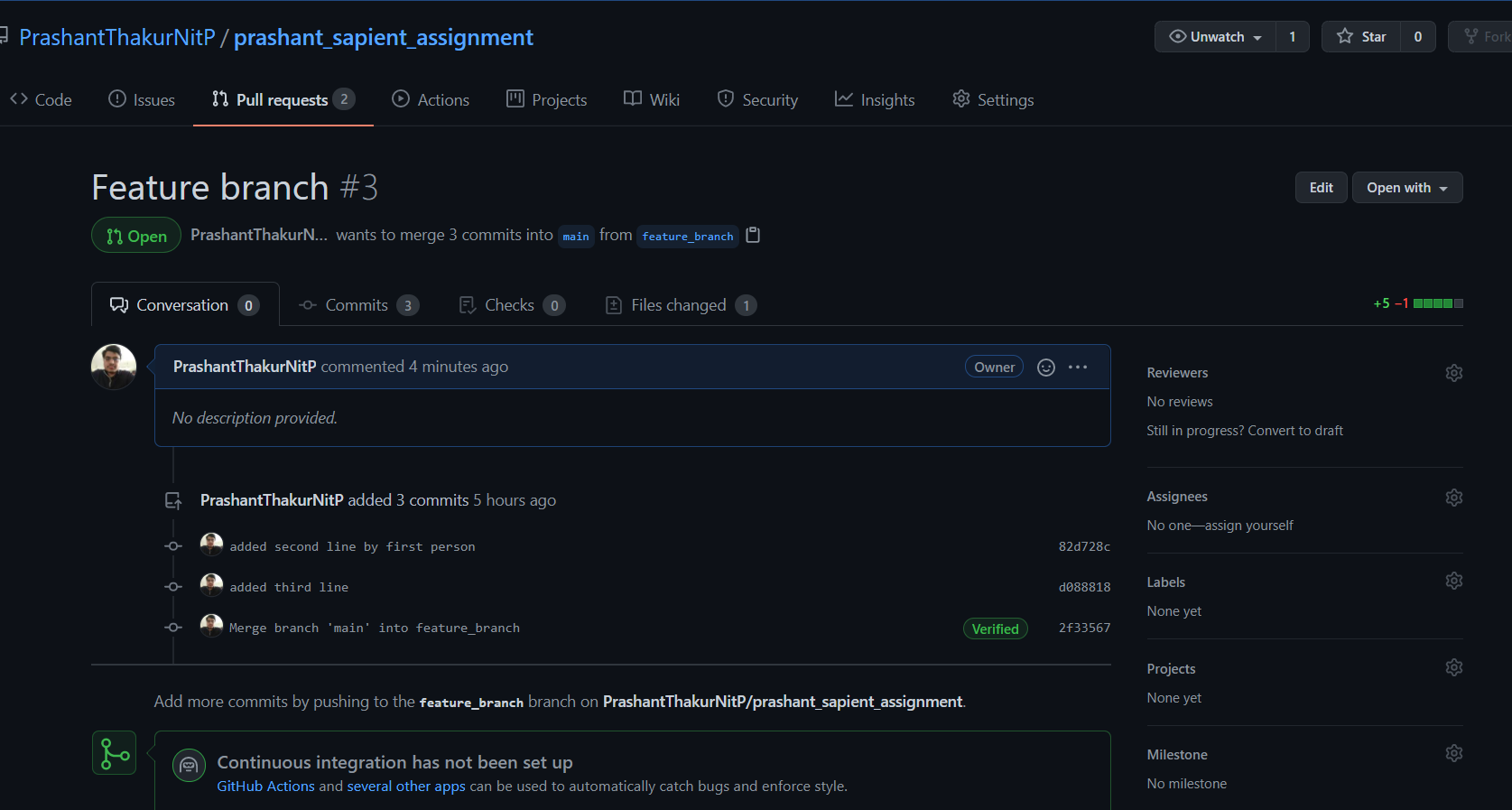


Solve conflict



-----------------------------------------------------------------------------------------------------------------

Now merge two branches



---------------------------------------------------------------------------------------------------------------------

**1 xi) Ensure the code is in sync. with the latest changes across off branches o from Procl. o QA o Dev. o Feature**

Ensure that code is in synchronization with latest changes actress all branch from prod, qa , dev ,feature

git checkout main

git rebase feature\_branch

git rebase qa

git rebase dev

git rebase dev

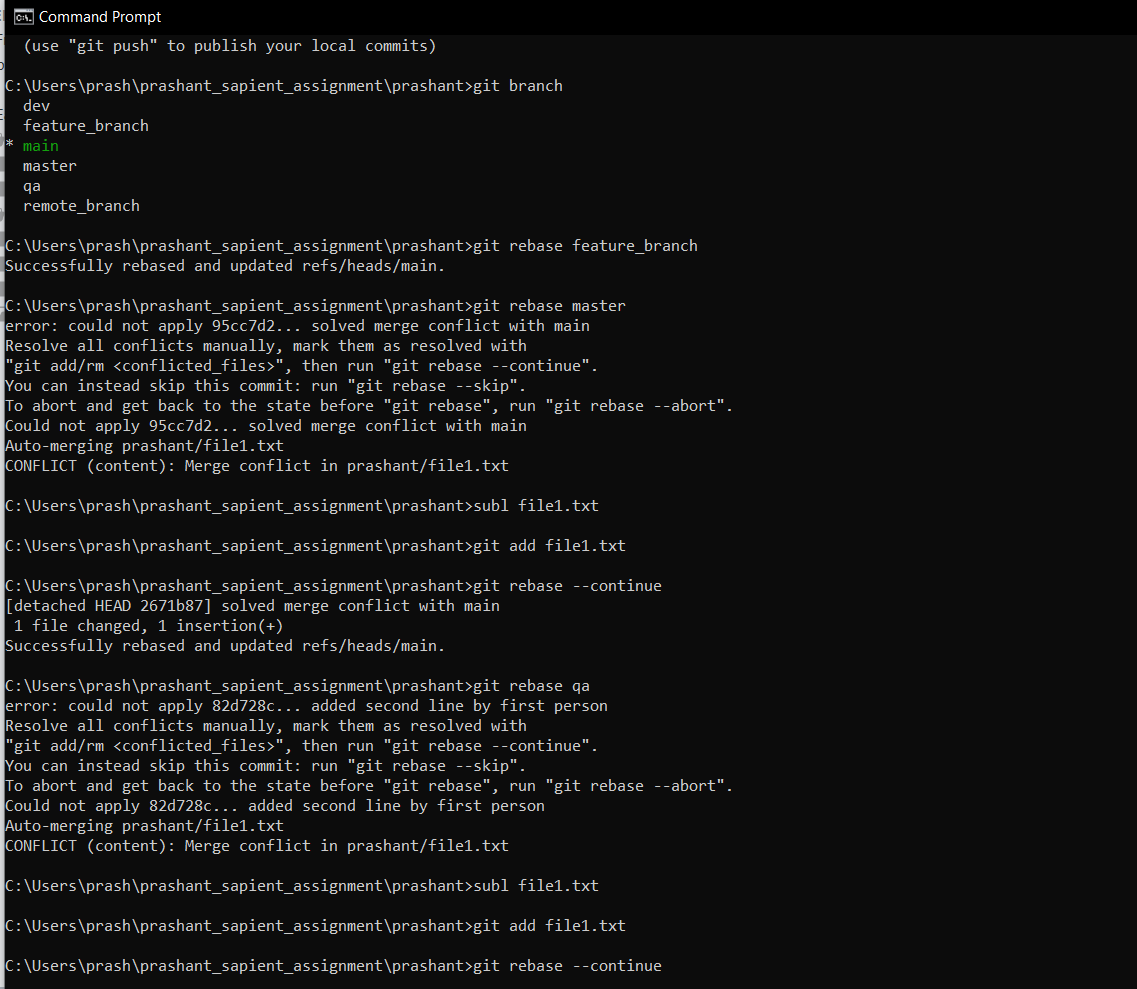
We can use rebase as well as merge option to make code synchronization with another

We are currently on main branch

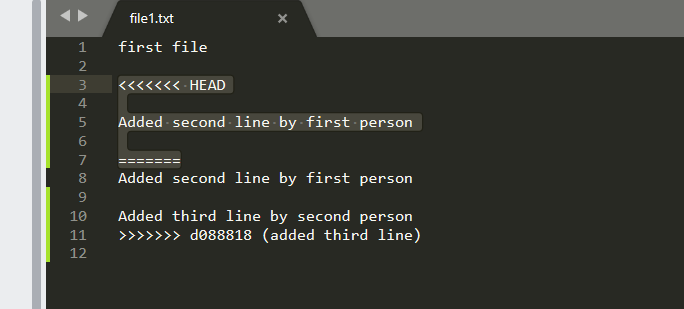
To make main branch updated we rebase other branches in main branch

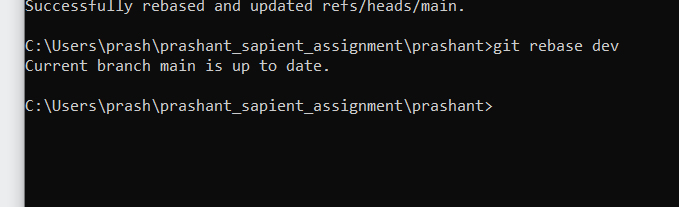
Thus we can create linear commit history and integrate code from other branches

We may need to solve conflict if it arises manully



Solving conflict that arise while rebasing





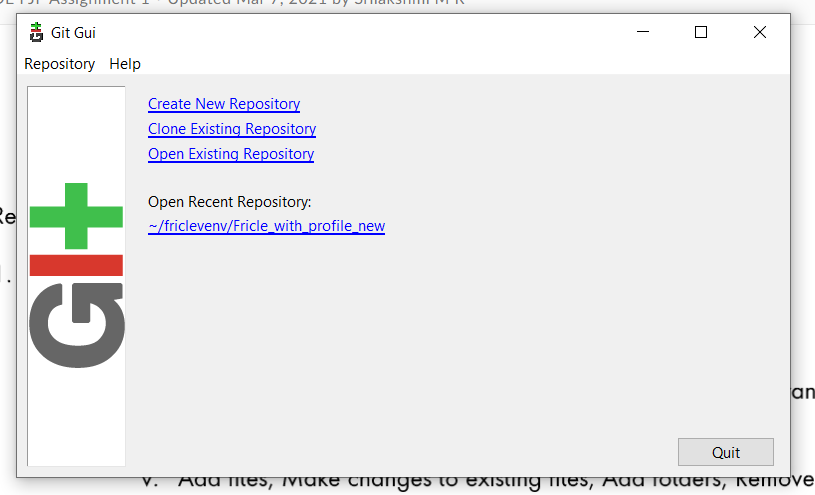
------------------------------------------------------------------------------------------------------------------------------

**Q1 b**

**a. Basic Usage using the GUI Client (VS Code or Git-hub Client)**

**Repeat all above steps using a GUI**

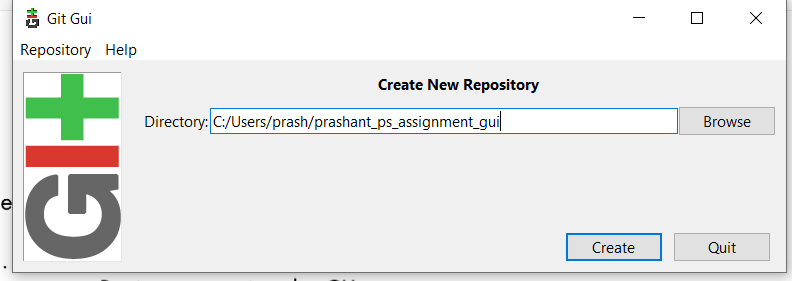
**-------------------------------------------------------------------------------------------------------------------------------**

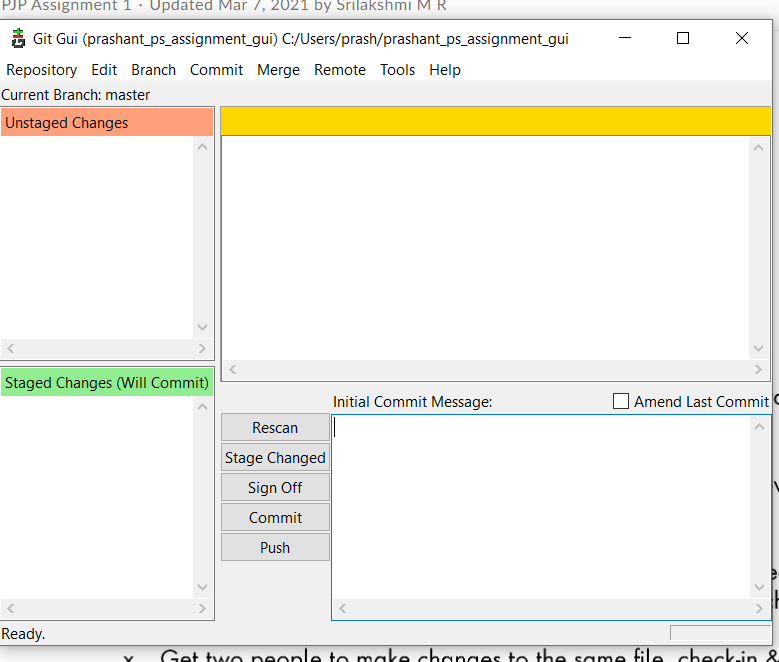
****

**Set up local repository**

**-----------------------------------------------------------------------------------------------------------------------------**

**Create a local repository**

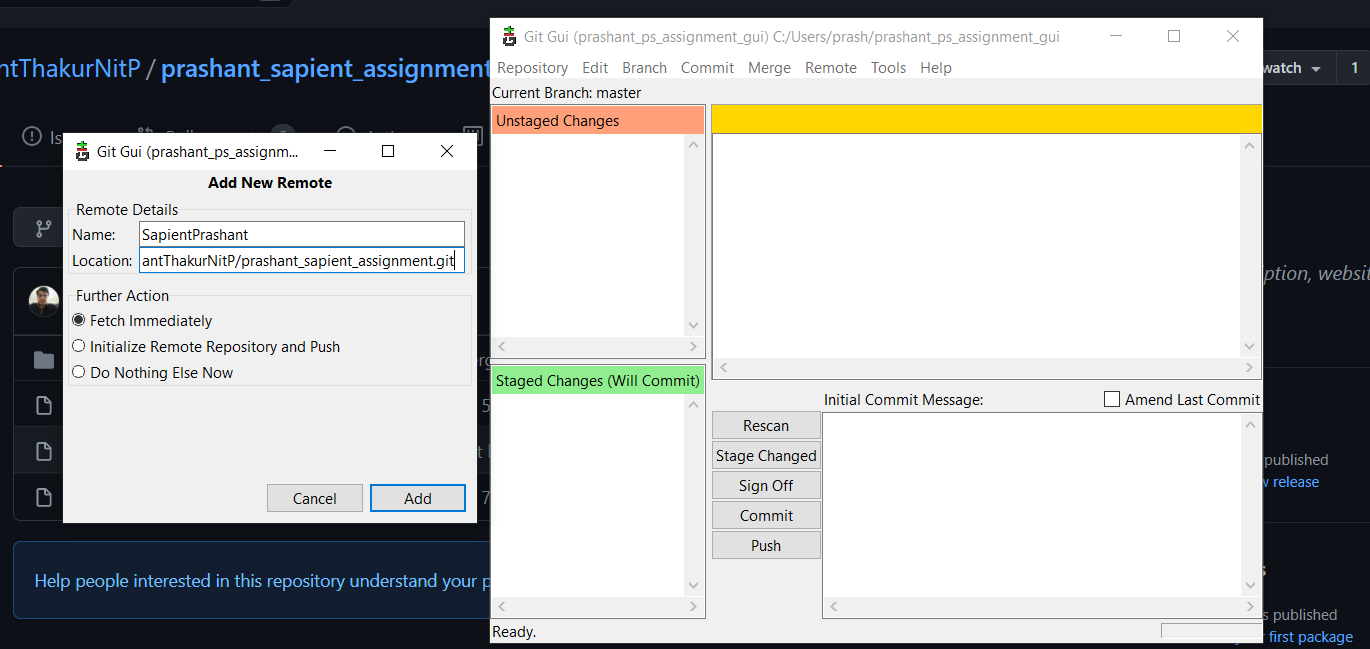
****

****

**Now we have successfully created a local repository . Now we will create a remote Github repository and set origin for this repository where we can push our code**

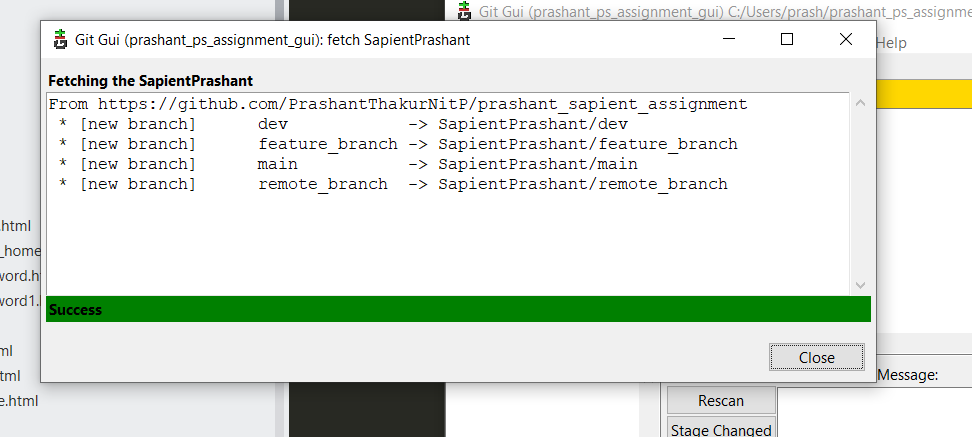
**-----------------------------------------------------------------------------------------------------------------------**

**Set up remote repository and set up origin**

****

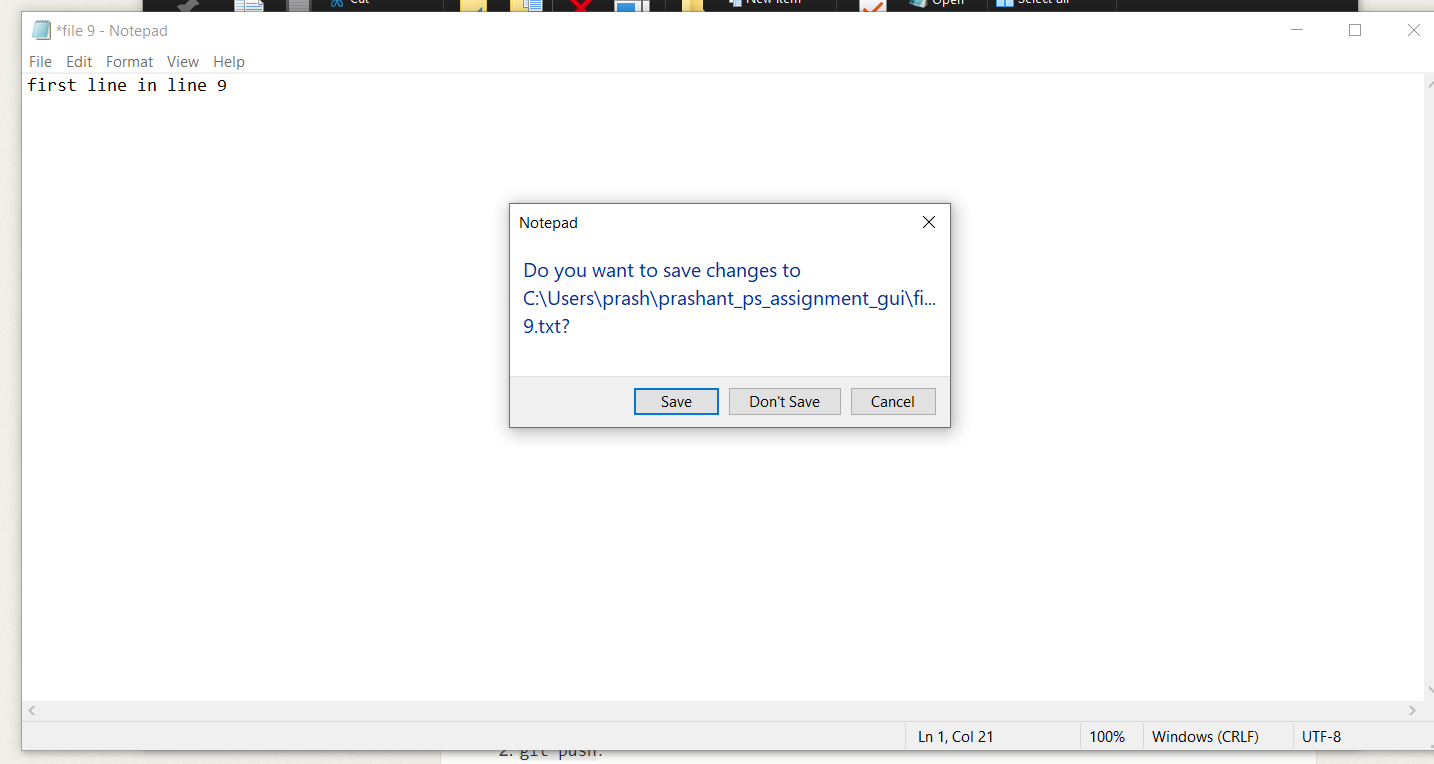
**-------------------------------------------------------------------------------------------------------------------------------**

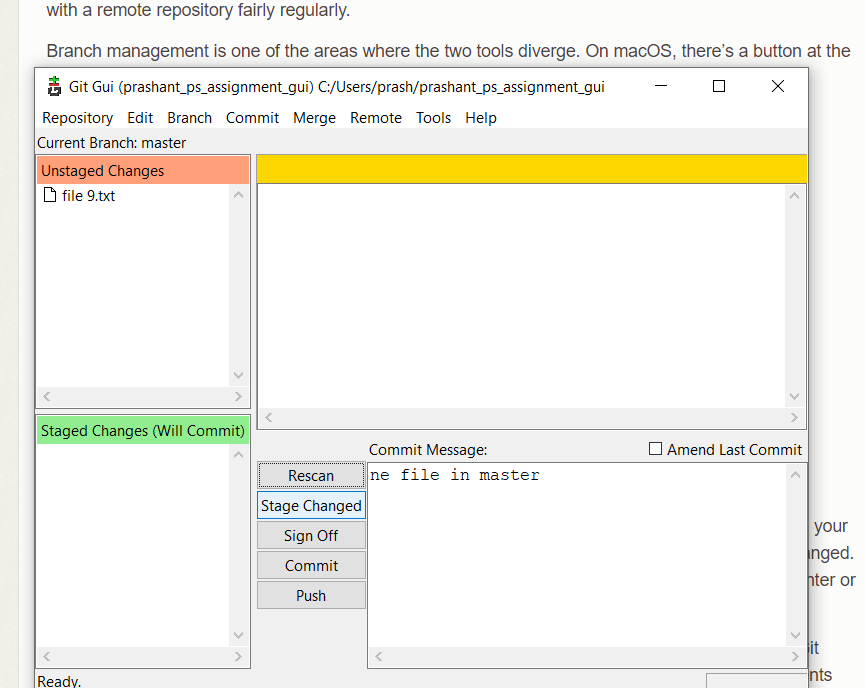
**Fetching from remote repository through pull**

****

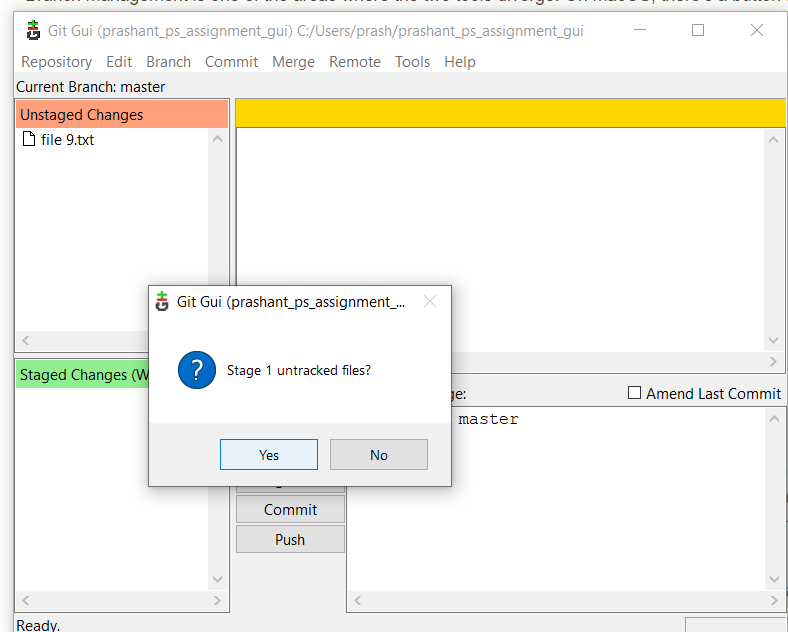
**---------------------------------------------------------------------------------------------------------------------------**

**Add a new file in cloned repository and save it**

****

****

**After saving file we add the file to staging area**

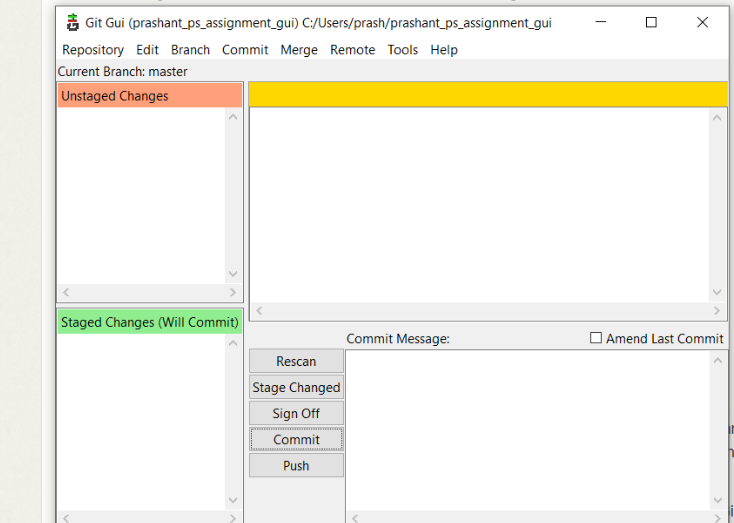
****

**Commit**

**After staging changes we need to commit it into local repository**

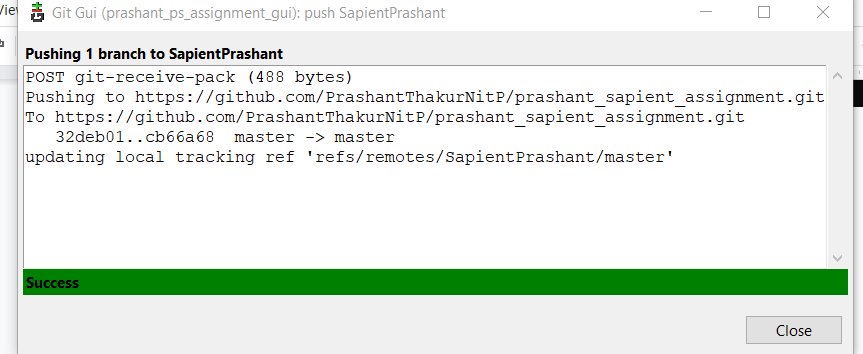
**We give commit message and then click on commit.**

**Our changes is commited in local github repository**

****

**Pushing code to remote branch**

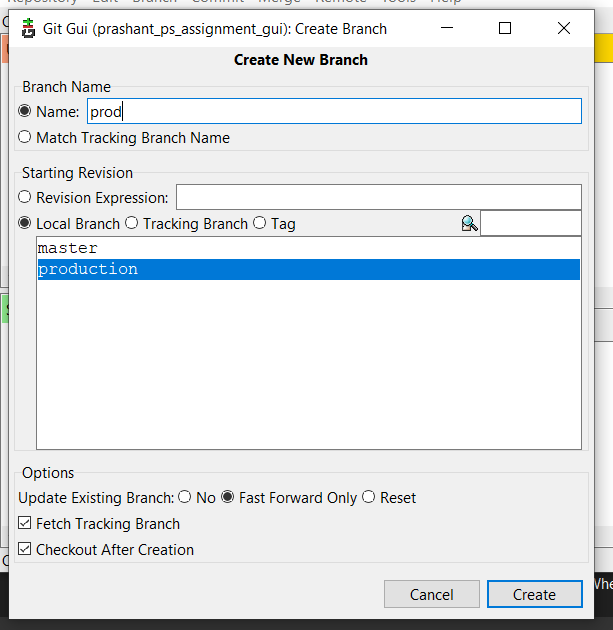
**Remote branch is Github repository where we will push our code after changes**

****

**Creating branche**

Branches help us to avoid conflict

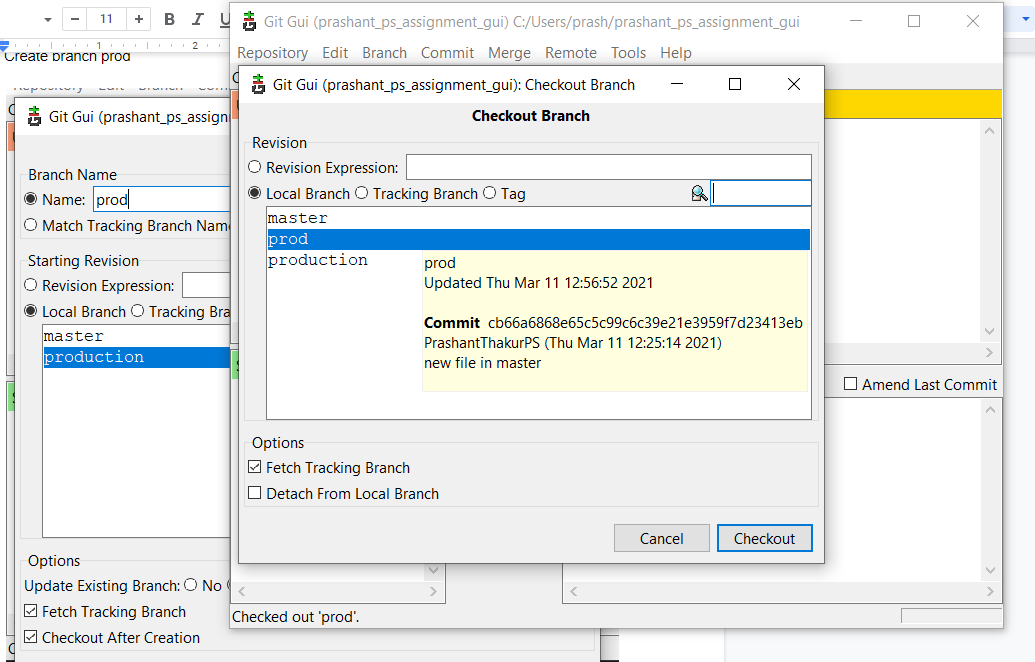
Create branch prod



**Checkout**

We can move from one local repository to another via checkout command

We can also create a new branch starting from current branch



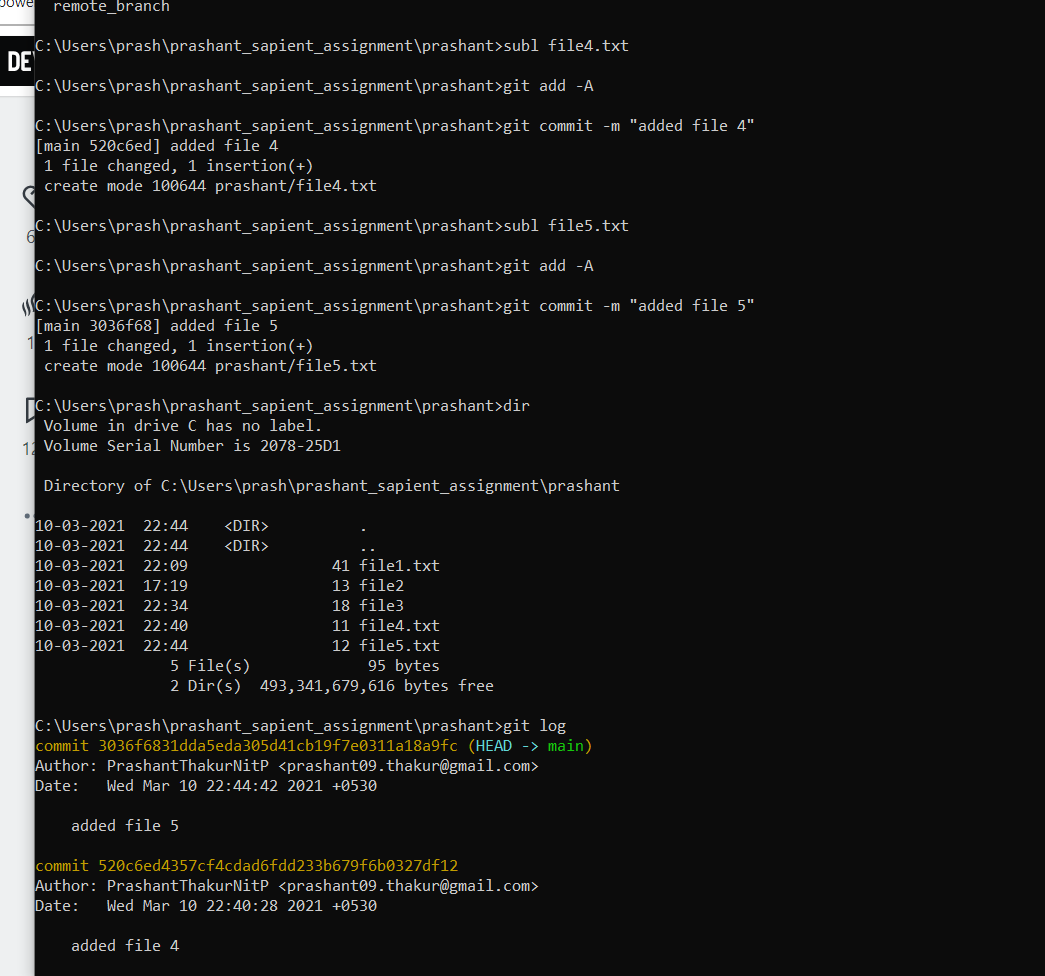
-----------------------------------------------------------------------------------------------------------------------

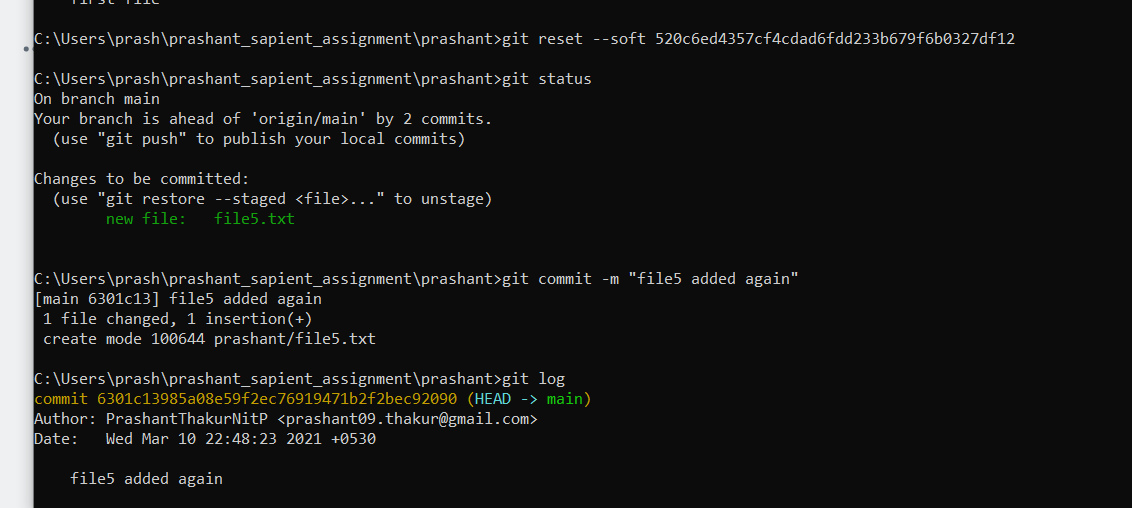
**Q1 c**

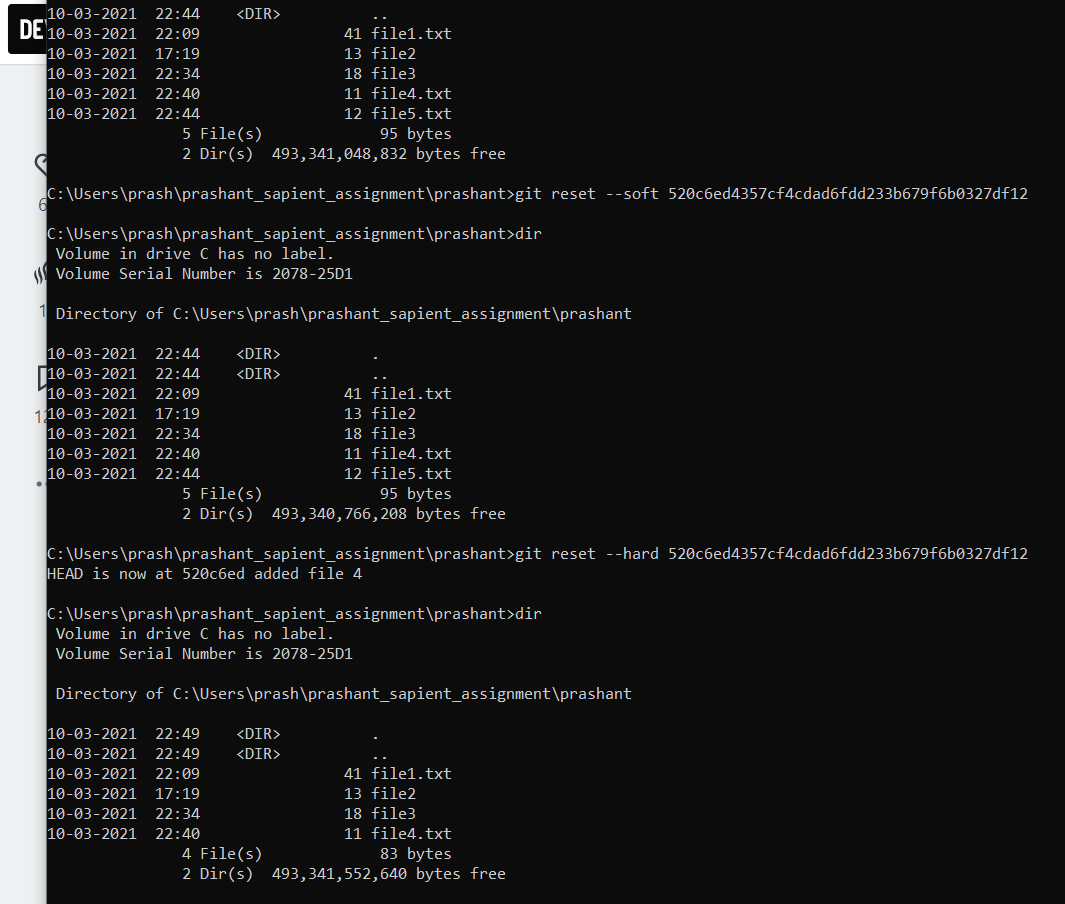
**i. Reset / revert one or more files to the previous state & ignore the locaI changes (Soft reset & Hard reset)**

Ans:

* Use this to return the entire working tree to the last committed state. This will discard commits in a private branch or throw away uncommitted changes!
* Changes which commit a branch HEAD is currently pointing at. It alters the existing commit history.
* Can be used to unstage a file
* --soft: Tells Git to reset HEAD to another commit, so index and the working directory will not be altered in any way. All of the files changed between the original HEAD and the commit will be staged.
* --hard: This resets everything - it resets HEAD back to another commit, resets the index to match it, and resets the working directory to match it as well.



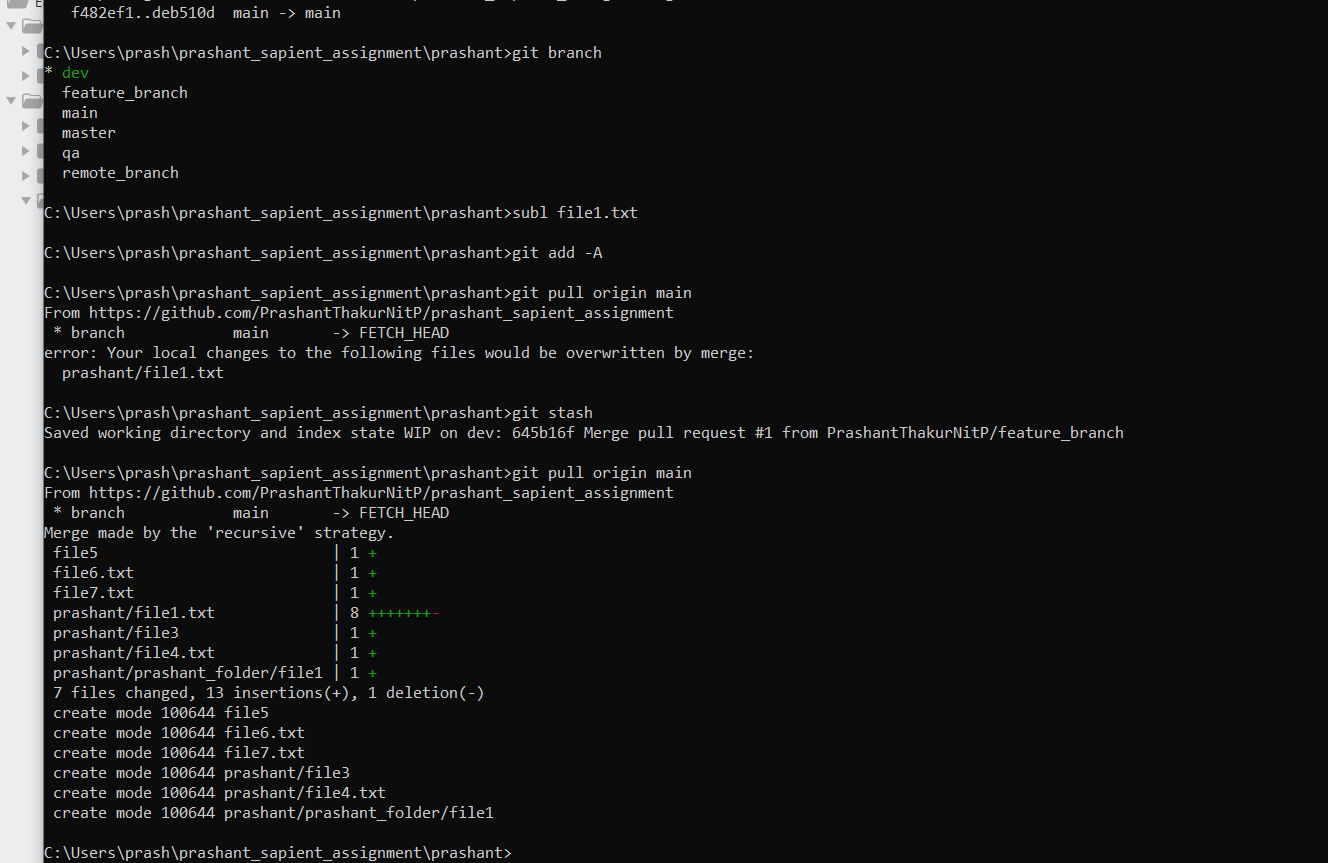




-----------------------------------------------------------------------------------------------------------------------

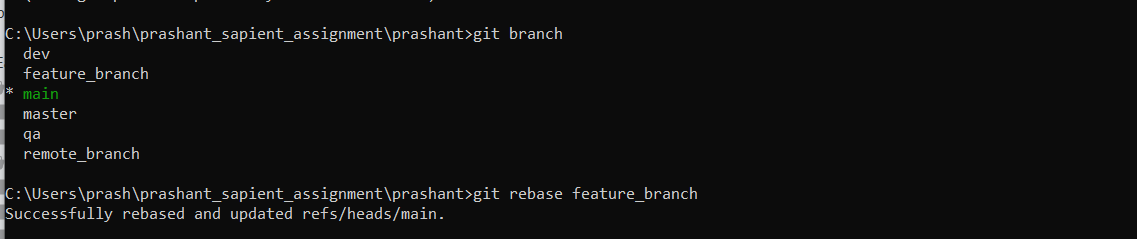
**1C**

**ii) Stash the focal changes during merge conflicts**



**1 C iii)**

**Rebasing with aIl options (reword, editt, squash, fixup, exec, drop )**

****

With the rebase command, you can take all the changes that were committed on one branch and replay them on a different branch.

"reword" allows us to change ONLY the commit message, NOT the commit contents

"edit" allows you to change BOTH commit contents AND commit message (the mechanism by which git allows you to edit the commit contents is by "pausing" the rebase; so you can amend the commit)

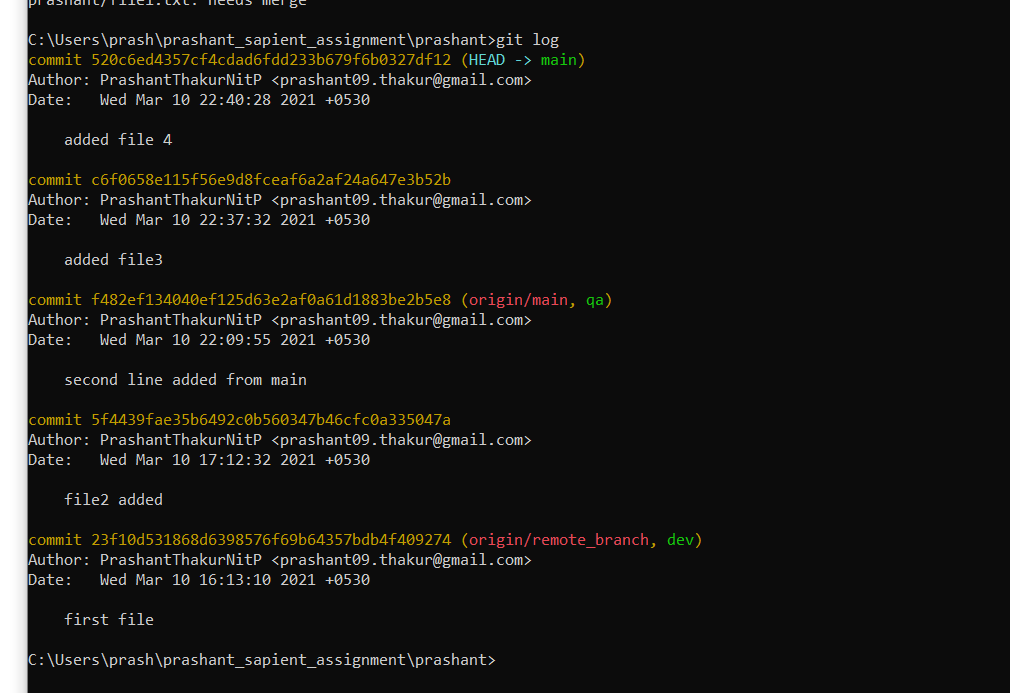
Two other commands rebase interactive offers us are:

squash (s for short), which melds the commit into the previous one (the one in the line before)

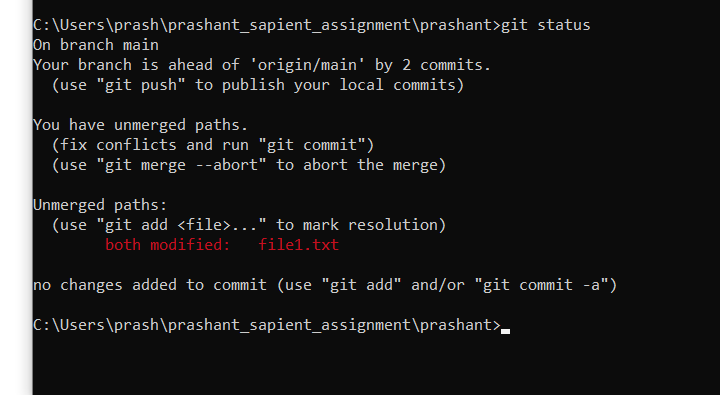
fixup (f for short), which acts like “squash”, but discards this commit’s message

**1C iv) Git fog, status & reflog**

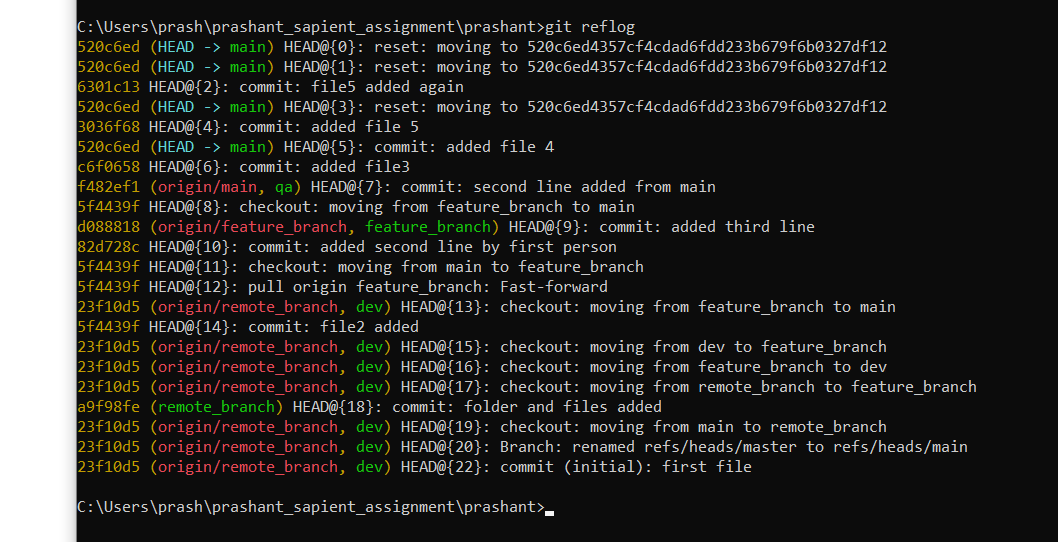
**git log**



**git status**



**git reflog**



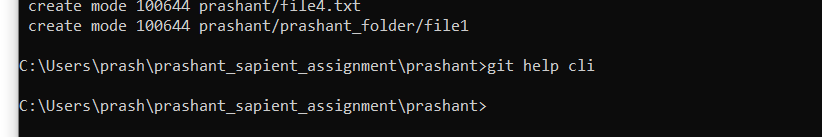
-------------------------------------------------------------------------------------------------------------------------------

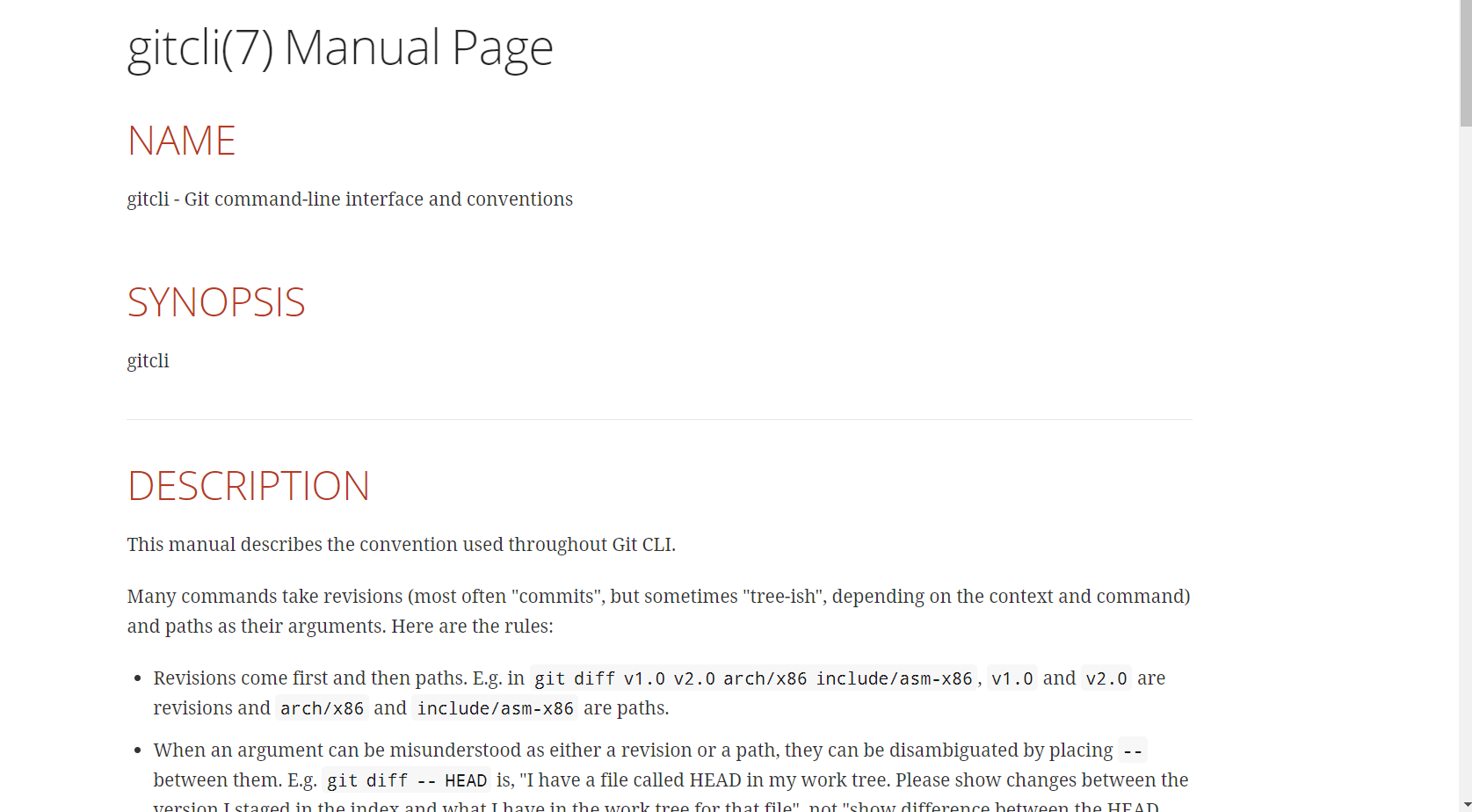
1 d)

# **a.** **Follow the assignments / ready made from git guides**

**Git command-fine interface and conventions**

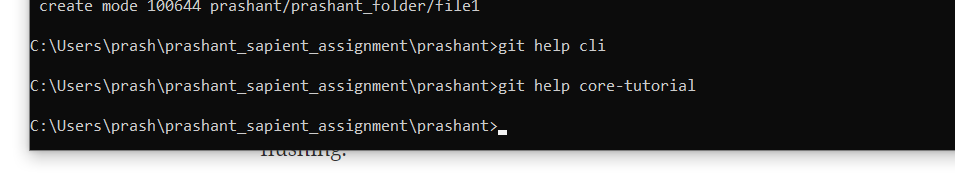
**git help cli**

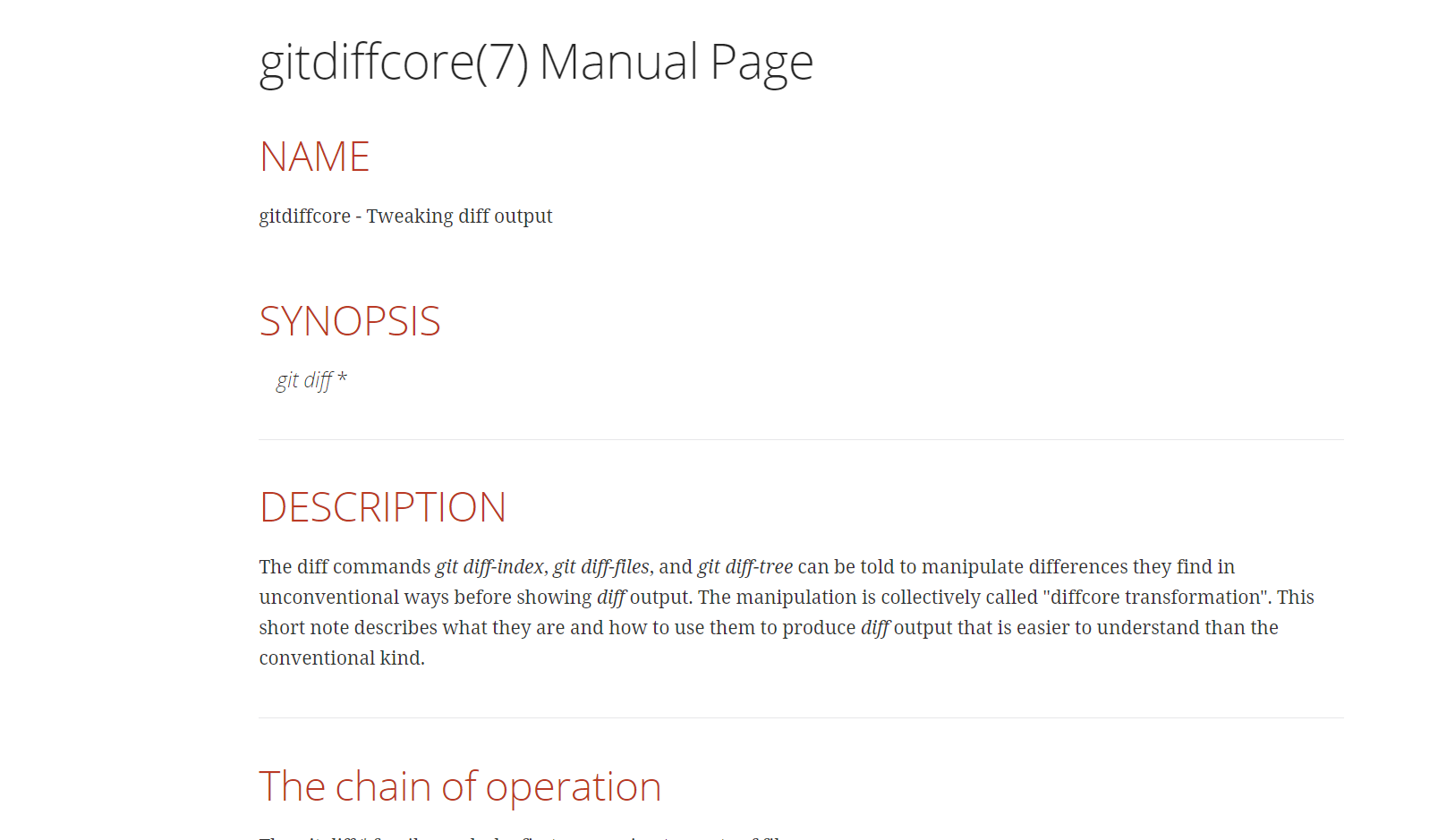




**i. A Git core tutorial for developers**

Git help core-tutorial

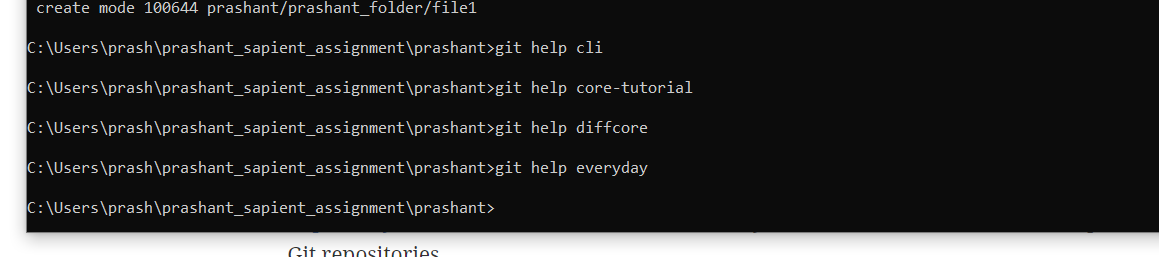


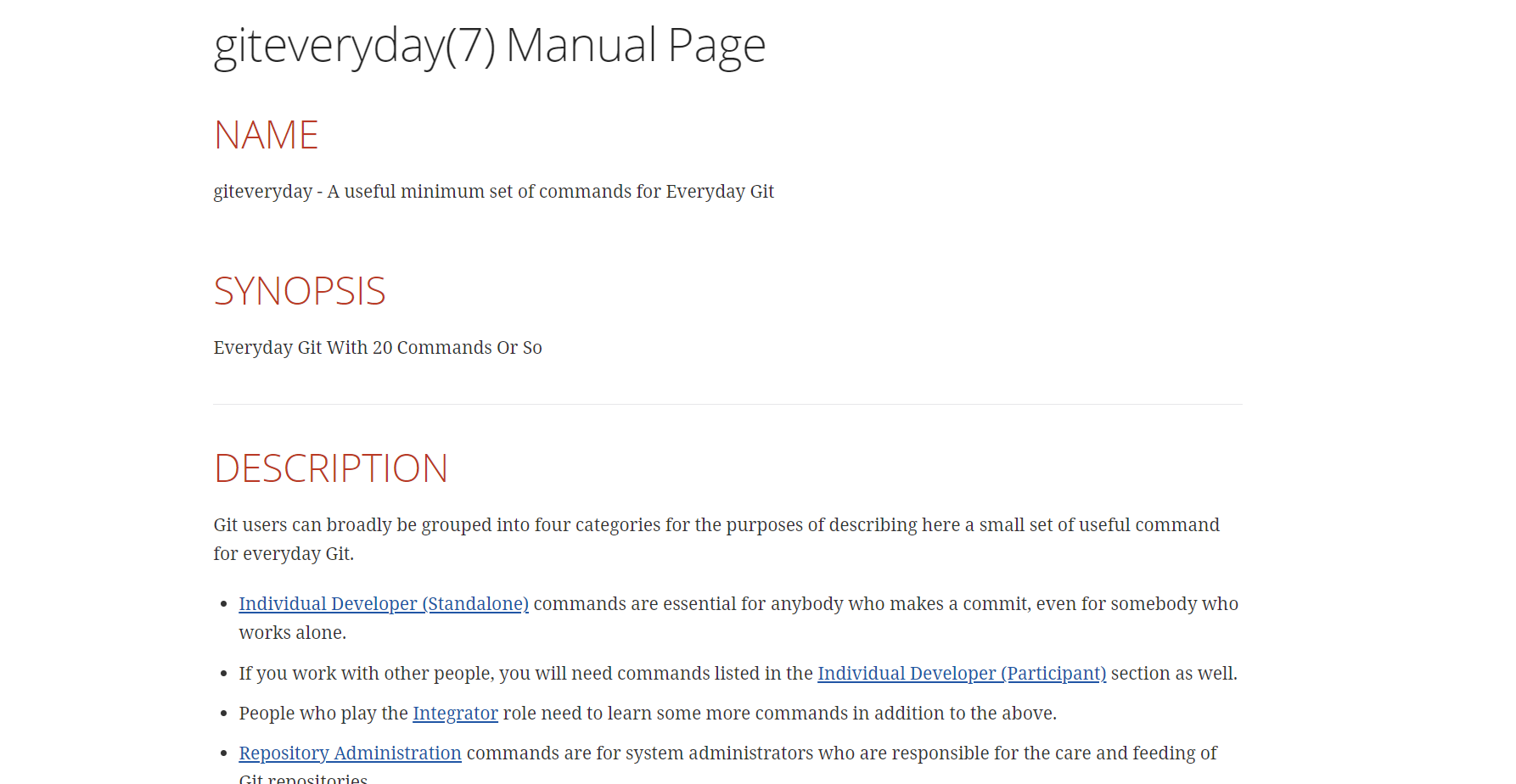


-------------------------------------------------------------------------------------------------------------------------------

**Tweaking diff output**

**A useful minimum set of commands for Everyday**





-------------------------------------------------------------------------------------------------------------------------------

**A Git Glossary**

**Specifies intentionally untracked files to ignore**

**Defining submodule properties**

**A tutorial introduction to Git**

**A tutorial introduction to Git: part two**

**An overview of recommended workflows with Git**

