**Q1: Git-hub:**

**(a): Basic usage using the CLI**

## **(1) Setup *a* Local Repository:**

## 

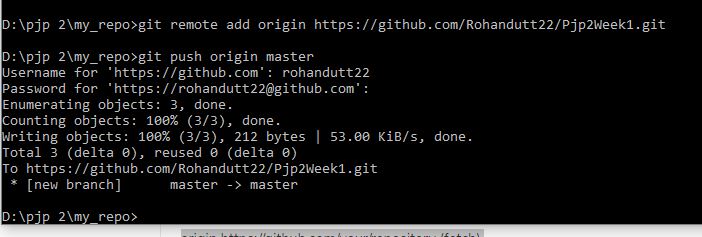
## **Sol:** git init my\_repo

## C:\Users\aser\Desktop\as\week1assn\1.1.JPG

**(2) Setup a Remote Repository:**

**Sol:** git remote add origin "http://github.com/Rohandutt22/Pjp2Week1.git"

git push origin master



**(3): Create Local branches (Feature branch, Dev. branch, QA Branch, Master / Prod. Branch, Delivery Branches)**

**Sol:** cd my\_repo

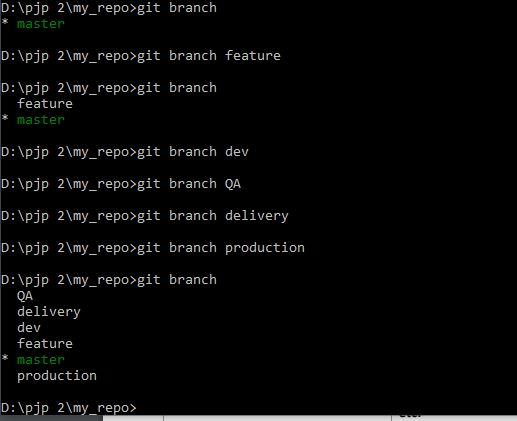
git branch feature

git branch QA

git branch dev

git branch delivery

git branch production



**(4): Create Remote branches**

**Sol:** git checkout feature

git push origin feature

git checkout QA

git push origin QA

git checkout dev

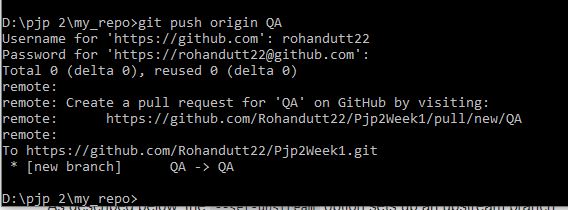
git push origin dev

git checkout delivery

git push origin delivery

git checkout production

git push origin production



**(5): Add files, Make changes to existing files, Add folders, Remove folders,**

**remove files**

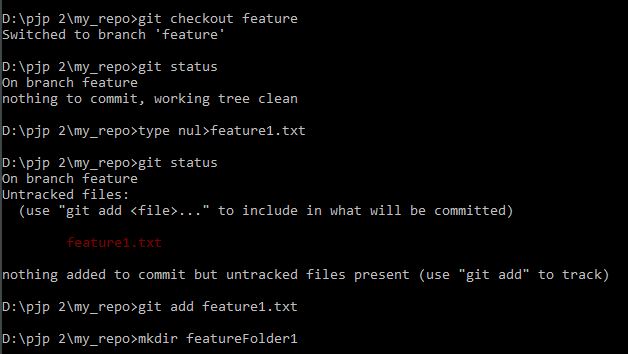
**Sol:** type nul>feature1.txt

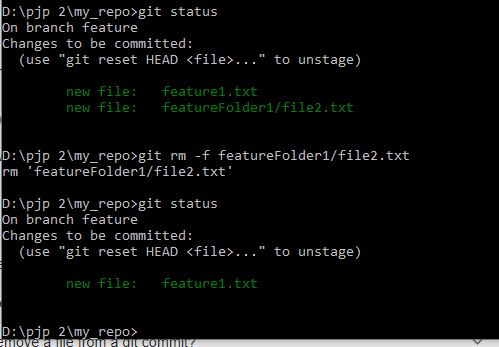
mkdir featureDirectory

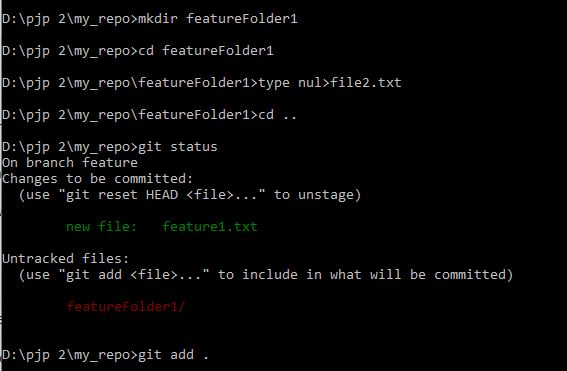
cd featureDirectory

type nul>file1.txt

git rm -f file1.txt







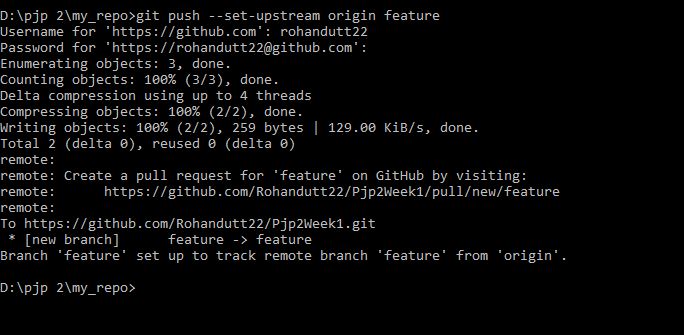
**(6): Check-in, Stage, Commit, Push files into Feature Branch**

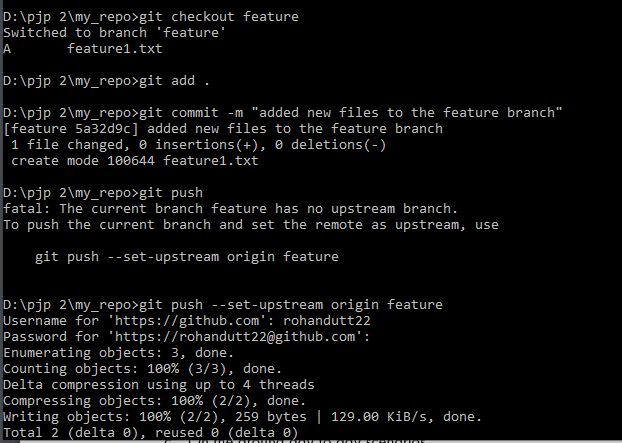
**Sol:** git checkout feature

git add .

git commit -m "files added to the feature branch"

git push

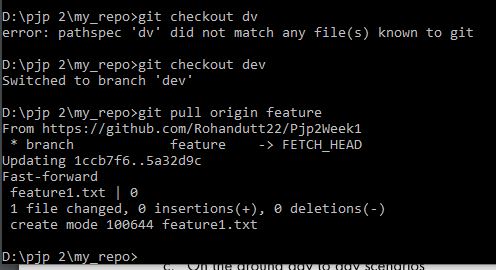




**(7): Promote code from Feature branch to Dev. branch via Pull requests**

**Sol:** git checkout dev

git pull origin feature



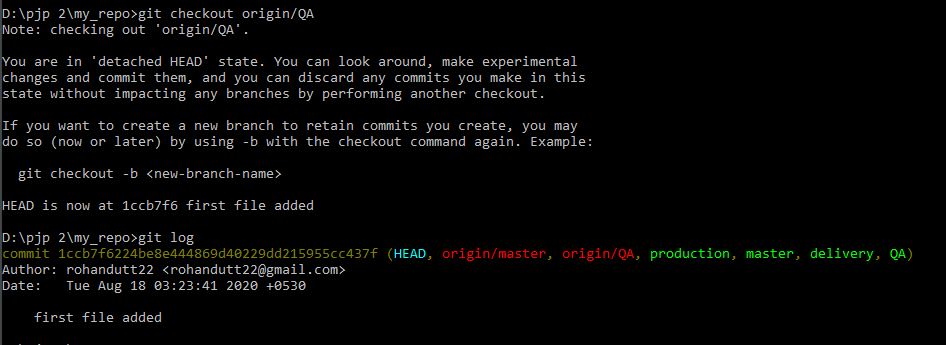
## **(8):** **Check-out the latest code from remote branch to local branch**

## **Sol:** git checkout production

## git branch -r (to get all the remote branches)

## git checkout origin/QA

## C:\Users\aser\Desktop\as\week1assn\1.8a.JPG



**(9): Explore the diff. between Checkout vs Pull**

**Sol:**

The basic difference between the checkout and pull is that in case of checkout we just copy or override the code from the branch on which we are checking out while in pull request we just update our code with the other code and our changes also remains there in the code.

# (10): Get two people to make changes to the same fife, checL-in & *handle* merge conflicts

**Sol:**

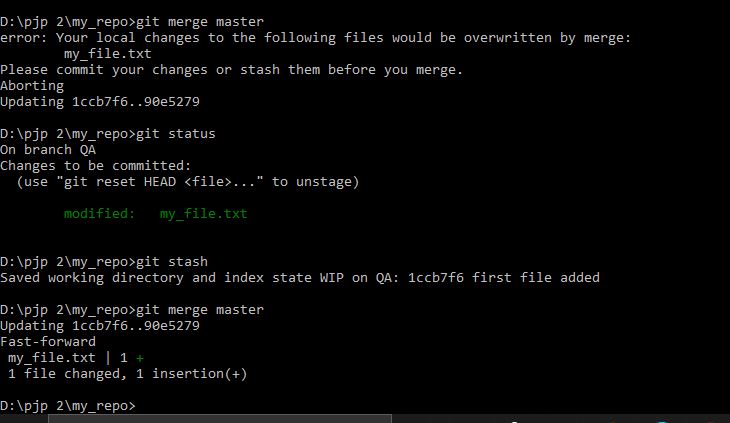
git checkout QA

(modify my\_file.txt,the other person will modify at master branch )

git merge master(wil show conflicts)

git stash

git merge master

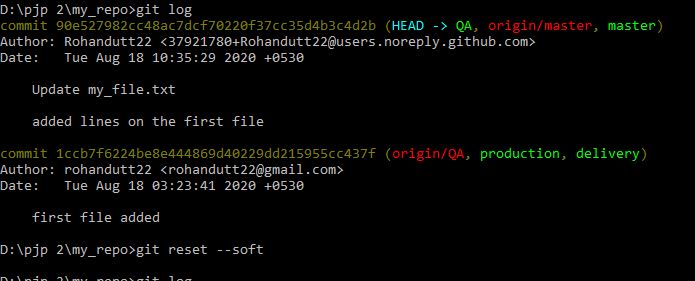


**C: On the ground day to day scenarios**

**(1): Reset / revert one or more files to the previous state & ignore the local changes (Soft reset & Hard reset)**

**Sol:** git log

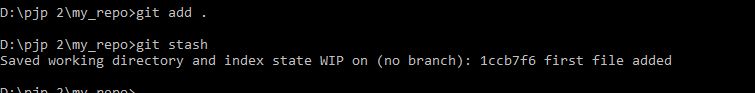
git reset –soft



**(2): Stash the local changes during merge conflicts**

**Sol:** git add .(to stage all the changes)

git stash



## **(3): Debasing with off options (reword, edit, squash, fixup, exec, drop)**

## **Sol:** git checkout delivery

## git rebase master

## C:\Users\aser\Desktop\as\week1assn\3.3.JPG

**(4): Git log, status & reffog**

## **Sol:**

## git status

## git add

## git reflog

## C:\Users\aser\Desktop\as\week1assn\3.4.JPG

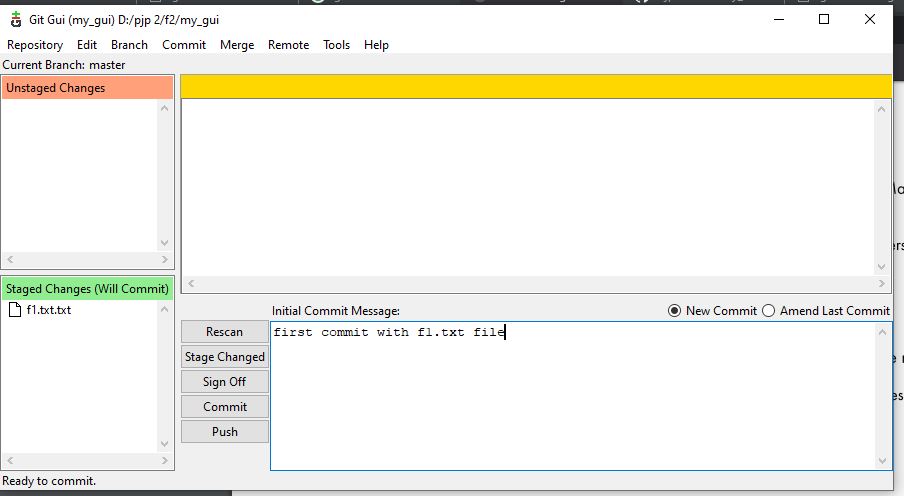
## (B)Basic usage using the GUI Client (VS Code or Github Client etc)

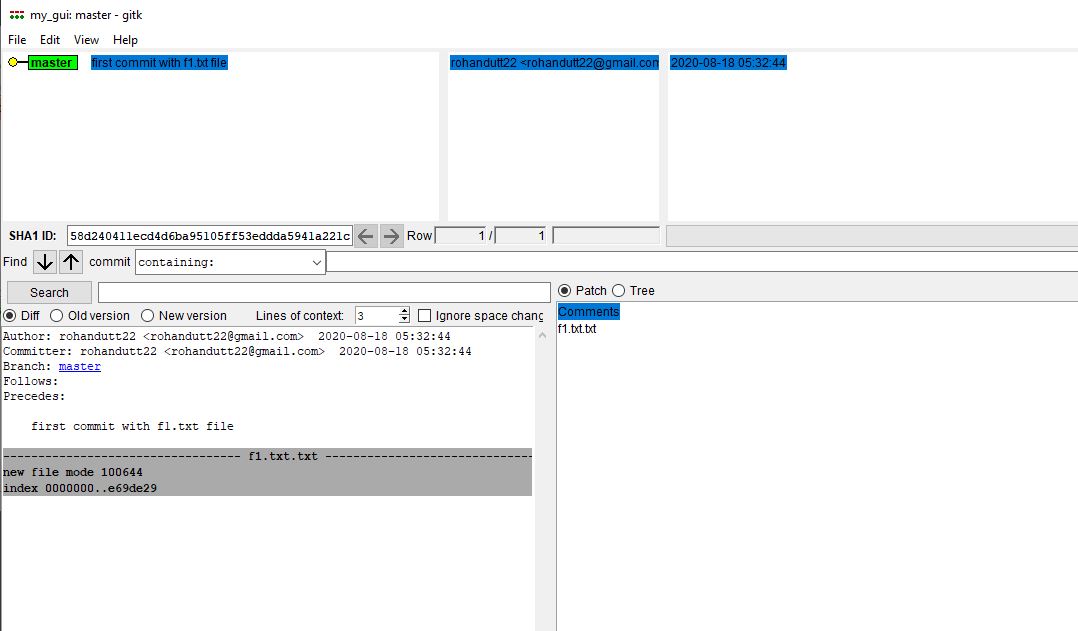
## **Sol:**

## C:\Users\aser\Desktop\as\week1assn\2.1a.JPG

## C:\Users\aser\Desktop\as\week1assn\2.3.JPG

## C:\Users\aser\Desktop\as\week1assn\2.4.JPG

****

****