Working with Git & Github

Let'c create one folder and initialized git repository to that folder which will create .git hidden folder

```
NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop
$ git -v
git version 2.41.0.windows.3

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop
$ mkdir project1

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop
$ cd project1

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1
$ echo 'Hello from first Project' >file.txt

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1
$ ls
file.txt

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1
$ cat file.txt
Hello from first Project

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1
$ git init
Initialized empty Git repository in C:/Users/NEW/Desktop/project1/.git/

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (master)
$ ls
file.txt

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (master)
$ ls -a
./  ../  .git/  file.txt
```

echo command to print some content > is used to add content to the mentioned file.
ls will list out files and directory inside the location.
ls -a flag will show all files and folders including hidden folders.
add command is used to add files in staging area and git status command is used to check tracked and untracked files

```
NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   file.txt


NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (master)
$ echo 'Another file'>file2.txt

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   file.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file2.txt
```

Let's Say by mistake you have stage that file.txt file then how to un stage. Now if you see the status both files are untacked.

```
NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (master)
$ git rm --cached file.txt
rm 'file.txt'

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file.txt
        file2.txt

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

I want to add both the files into staging area for that you can mention

git add file.txt file2.txt (this also works) but what if you have multiple files then we can use . it will add all the untracked files into staging area.

```
NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (master)
$ git add .

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   file.txt
        new file:   file2.txt
```

To add this staged files into Local Repository we can use git commit command.

ls -l command for getting files with all details like permissions, sizes, time or creation etc..

```
NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (master)
$ git commit -m 'Feature X completed'
[master (root-commit) 90e30fa] Feature X completed
 2 files changed, 2 insertions(+)
 create mode 100644 file.txt
 create mode 100644 file2.txt

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (master)
$ ls -l
total 2
-rw-r--r-- 1 NEW 197121 25 Feb 11 15:32 file.txt
-rw-r--r-- 1 NEW 197121 13 Feb 11 15:39 file2.txt
```

Let's See the commit details using log command.

```
NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (master)
$ git log
commit 90e30fa8b4937f40e9a83a1205b1f67a2d4d757a (HEAD -> master)
Author: sonam-niit <sonam.gravity@gmail.com>
Date:   Tue Feb 11 15:51:06 2025 +0530

    Feature X completed

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (master)
$ git log --oneline
90e30fa (HEAD -> master) Feature X completed
```

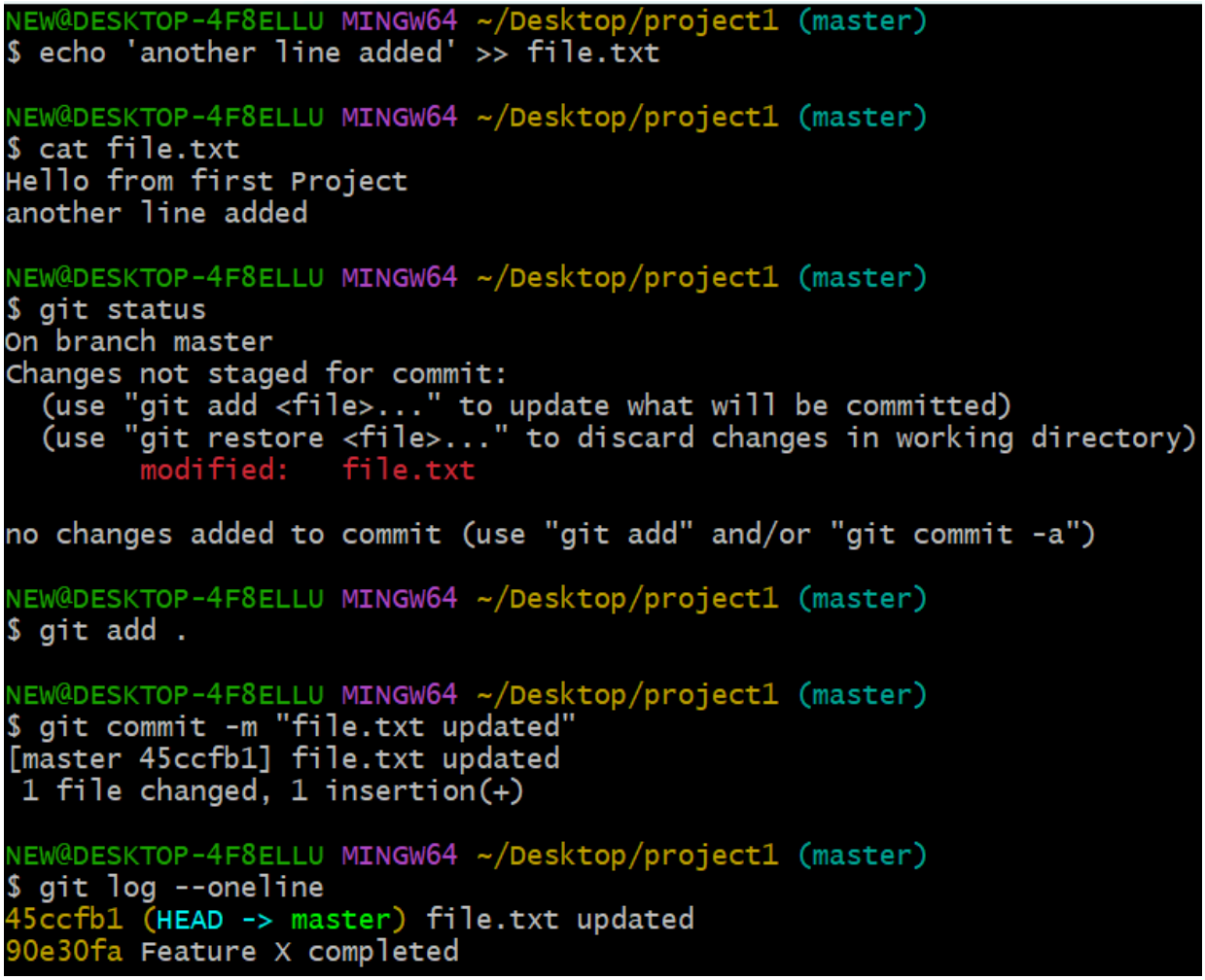
I want to make some changes to file.txt. I recreated file.txt

```
NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (master)
$ echo 'Another line added'>file.txt

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (master)
$ cat file.txt
Another line added
```

I lost the committed code, to get that code back we can execute restore command

```
NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (master)
$ git restore file.txt

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (master)
$ cat file.txt
Hello from first Project
```

Let's update the file.txt, stage it and commit it.



The latest commit we did by mistake

To resolve issue we have 2 option hard reset and soft reset.

If you do just reset by default its Soft Reset means it will keep changes in working directory but removed from staging and local repo.

If you do hard reset it will remove the changes from commit as well as from working directory.

```
NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (master)
$ git reset 90e30fa
Unstaged changes after reset:
M       file.txt

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (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:   file.txt

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

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (master)
$ git log
commit 90e30fa8b4937f40e9a83a1205b1f67a2d4d757a (HEAD -> master)
Author: sonam-niit <sonam.gravity@gmail.com>
Date:   Tue Feb 11 15:51:06 2025 +0530

    Feature X completed
```

Now let do the same process for Hard Reset.

```
NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (master)
$ git add .

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (master)
$ git commit -m "File2 updated"
[master 84d6655] File2 updated
 1 file changed, 1 insertion(+)

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (master)
$ git log --oneline
84d6655 (HEAD -> master) File2 updated
90e30fa Feature X completed

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (master)
$ git reset 90e30fa --hard
HEAD is now at 90e30fa Feature X completed

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (master)
$ git log
commit 90e30fa8b4937f40e9a83a1205b1f67a2d4d757a (HEAD -> master)
Author: sonam-niit <sonam.gravity@gmail.com>
Date:   Tue Feb 11 15:51:06 2025 +0530

    Feature X completed

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (master)
$ git status
On branch master
nothing to commit, working tree clean
```

We can do restore from staging area as well.

```
NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (master)
$ echo 'New File added'>test.txt

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (master)
$ git add test.txt

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   test.txt


NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (master)
$ rm test.txt

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (master)
$ ls
file.txt  file2.txt

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (master)
$ git restore test.txt

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (master)
$ ls
file.txt  file2.txt  test.txt
```

Git Configuration

```
NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (master)
$ git config --global user.name "sonam-niit"

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (master)
$ git config --global user.email "sonam.gravity@gmail.com"

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (master)
$ git config -l --global
user.email=sonam.gravity@gmail.com
user.name=sonam-niit
```

Create one Repository on Github

execute the command to set main as main Branch.

```
NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (master)
$ git branch -M main
```

Set your repository origin and push code to github

```
NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (main)
$ git remote add origin https://github.com/sonam-niit/sample-git.git

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (main)
$ git push -u origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (7/7), 520 bytes | 173.00 KiB/s, done.
Total 7 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/sonam-niit/sample-git.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

For the very first time it will pop up with browser authentication, you need to add your credentials and get authenticated then it will push code to remote repo.

Go to your repository click on add read me file, you can add some content to it and commit with some message directly from browser.

Now to get your local repository up to date execute git pull command.

```
NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (main)
$ git pull
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 1007 bytes | 77.00 KiB/s, done.
From https://github.com/sonam-niit/sample-git
   d038212..0d868ae  main       -> origin/main
Merge made by the 'ort' strategy.
 README.md | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
```

If you try to make some local changes without pull and from local if you try to push you can get the error as below. So here you can solve the issue by doing git pull and then do git push.

```
NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/project1 (main)
$ git push -u origin main
To https://github.com/sonam-niit/sample-git.git
 ! [rejected]        main -> main (fetch first)
error: failed to push some refs to 'https://github.com/sonam-niit/sample-git.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```