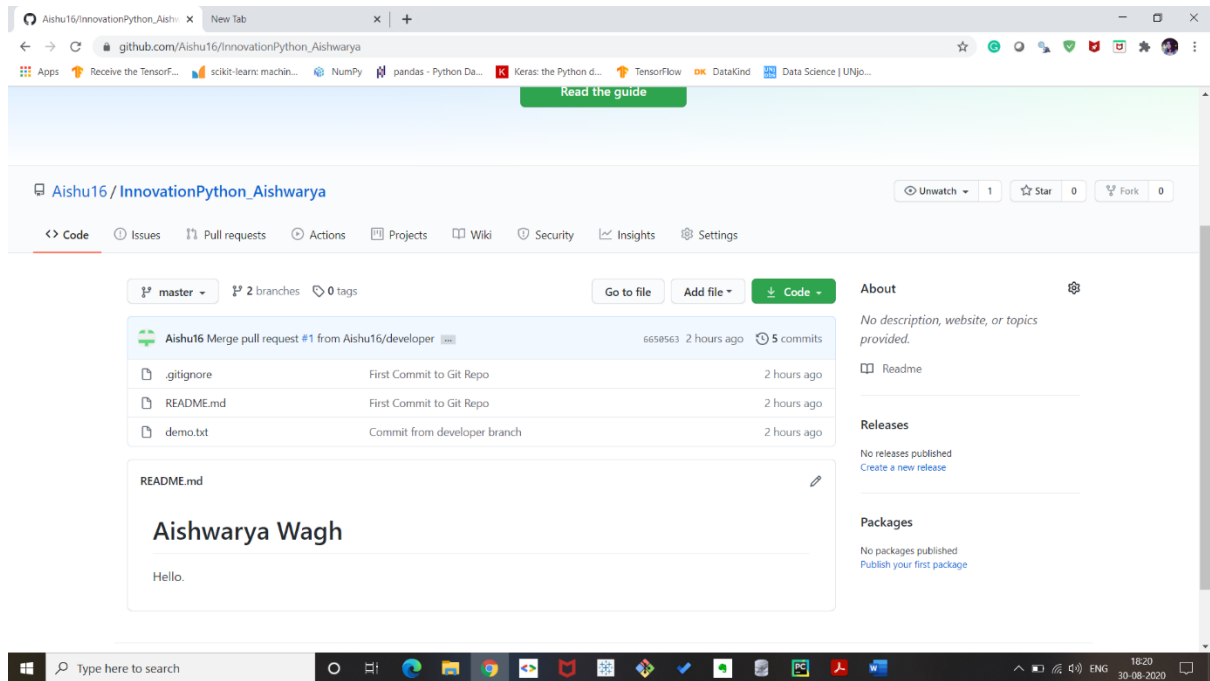


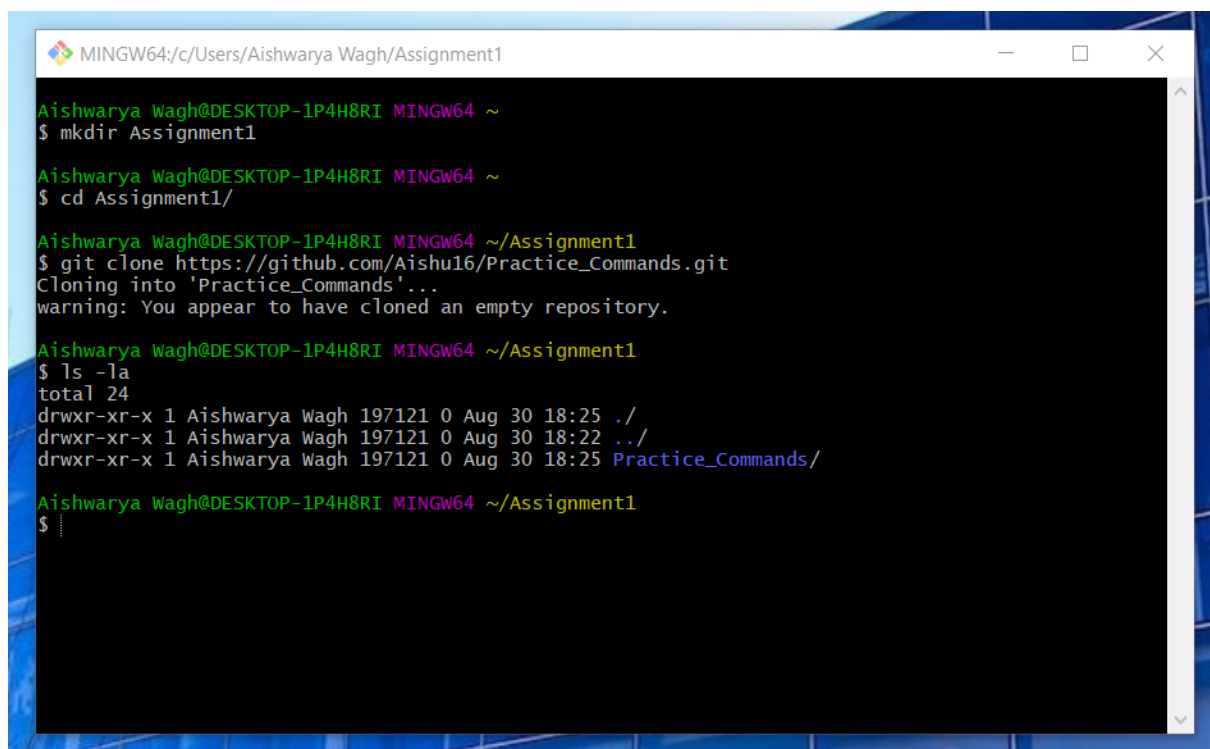
Git Tasks

1. Make a repository on GitHub with the name “**InnovationPython_yourname**”
eg: “**InnovationPython_Ankush**”.



a. Practice on following commands:

■ Git Clone



■ Git Diff

- Git Add
- Git Status

```
MINGW64:/c/Users/Aishwarya Wagh/Assignment1/Practice_Commands

Aishwarya Wagh@DESKTOP-1P4H8RI MINGW64 ~/Assignment1/Practice_Commands (master)
$ ls -la
total 9
drwxr-xr-x 1 Aishwarya Wagh 197121 0 Aug 30 18:40 ./
drwxr-xr-x 1 Aishwarya Wagh 197121 0 Aug 30 18:35 ../
drwxr-xr-x 1 Aishwarya Wagh 197121 0 Aug 30 18:25 .git/
-rw-r--r-- 1 Aishwarya Wagh 197121 25 Aug 30 18:38 Practice.txt

Aishwarya Wagh@DESKTOP-1P4H8RI MINGW64 ~/Assignment1/Practice_Commands (master)
$ git add .

Aishwarya Wagh@DESKTOP-1P4H8RI MINGW64 ~/Assignment1/Practice_Commands (master)
$ git status
On branch master

No commits yet

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

        new file:   Practice.txt

Aishwarya Wagh@DESKTOP-1P4H8RI MINGW64 ~/Assignment1/Practice_Commands (master)
$ |
```

- Git Log

```
MINGW64:/c/Users/Aishwarya Wagh/Assignment1/Practice_Commands

Aishwarya Wagh@DESKTOP-1P4H8RI MINGW64 ~/Assignment1/Practice_Commands (master)
$ git log
commit c4d539e89708c9d5529494b0085b760360b191aa (HEAD -> master)
Author: aishwarya <aishwaryawagh94@gmail.com>
Date:   Sun Aug 30 18:45:53 2020 -0400

    first Commit in Practice command

Aishwarya Wagh@DESKTOP-1P4H8RI MINGW64 ~/Assignment1/Practice_Commands (master)
$
```

- Git Commit

```
MINGW64/c/Users/Aishwarya Wagh/Assignment1/Practice_Commands

Aishwarya Wagh@DESKTOP-1P4H8RI MINGW64 ~/Assignment1/Practice_Commands (master)
$ git commit -m "first Commit in Practice command"
[master (root-commit) c4d539e] first Commit in Practice command
1 file changed, 2 insertions(+)
create mode 100644 Practice.txt

Aishwarya Wagh@DESKTOP-1P4H8RI MINGW64 ~/Assignment1/Practice_Commands (master)
$
```

■ Git Push

```
MINGW64/c/Users/Aishwarya Wagh/Assignment1/Practice_Commands

Aishwarya Wagh@DESKTOP-1P4H8RI MINGW64 ~/Assignment1/Practice_Commands (master)
$ git push origin master
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 565 bytes | 282.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0)
To https://github.com/Aishu16/Practice_Commands.git
   8aeb20..303d1ce master -> master

Aishwarya Wagh@DESKTOP-1P4H8RI MINGW64 ~/Assignment1/Practice_Commands (master)
$ |
```

■ Git Reset

```
MINGW64:/c/Users/Aishwarya Wagh/Assignment1/Practice Commands
Aishwarya Wagh@DESKTOP-1P4H8RI MINGW64 ~/Assignment1/Practice_Commands (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

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

    Reset_Command_Practice.txt

nothing added to commit but untracked files present (use "git add" to track)
Aishwarya Wagh@DESKTOP-1P4H8RI MINGW64 ~/Assignment1/Practice_Commands (master)
$ git add Reset_Command_Practice.txt
Aishwarya Wagh@DESKTOP-1P4H8RI MINGW64 ~/Assignment1/Practice_Commands (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   Reset Command Practice.txt

Aishwarya Wagh@DESKTOP-1P4H8RI MINGW64 ~/Assignment1/Practice_Commands (master)
$ git reset HEAD Reset_Command_Practice.txt
Aishwarya Wagh@DESKTOP-1P4H8RI MINGW64 ~/Assignment1/Practice_Commands (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

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

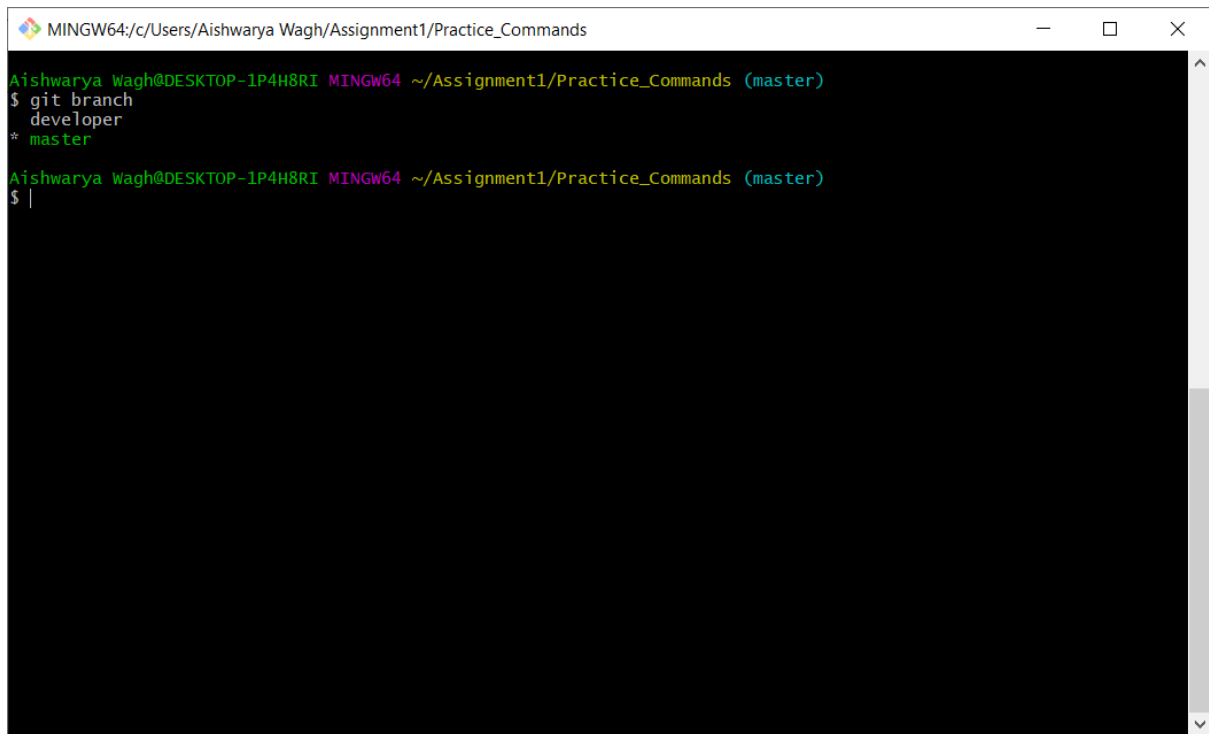
    Reset_Command_Practice.txt

nothing added to commit but untracked files present (use "git add" to track)
Aishwarya Wagh@DESKTOP-1P4H8RI MINGW64 ~/Assignment1/Practice_Commands (master)
$ |
```

■ Git Pull

```
MINGW64:/c/Users/Aishwarya Wagh/Assignment1/Practice_Commands
Aishwarya Wagh@DESKTOP-1P4H8RI MINGW64 ~/Assignment1/Practice_Commands (master)
$ git pull origin master --allow-unrelated-histories
From https://github.com/Aishu16/Practice_Commands
 * branch      master       -> FETCH_HEAD
Merge made by the 'recursive' strategy.
 README.md | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
Aishwarya Wagh@DESKTOP-1P4H8RI MINGW64 ~/Assignment1/Practice_Commands (master)
$ |
```

■ Git Branch

A terminal window titled 'MINGW64:/c/Users/Aishwarya Wagh/Assignment1/Practice_Commands'. The prompt is 'Aishwarya Wagh@DESKTOP-1P4H8RI MINGW64 ~/Assignment1/Practice_Commands (master)'. The command '\$ git branch' is entered, and the output is 'developer' and '* master'.

```
MINGW64:/c/Users/Aishwarya Wagh/Assignment1/Practice_Commands
Aishwarya Wagh@DESKTOP-1P4H8RI MINGW64 ~/Assignment1/Practice_Commands (master)
$ git branch
  developer
* master
Aishwarya Wagh@DESKTOP-1P4H8RI MINGW64 ~/Assignment1/Practice_Commands (master)
$ |
```

■ Git Checkout

A terminal window titled 'MINGW64:/c/Users/Aishwarya Wagh/Assignment1/Practice_Commands'. The prompt is 'Aishwarya Wagh@DESKTOP-1P4H8RI MINGW64 ~/Assignment1/Practice_Commands (master)'. The command '\$ git checkout developer' is entered, and the output is 'Switched to branch 'developer''. The prompt changes to '(developer)'. Then, the command '\$ git branch' is entered, and the output is '* developer' and 'master'.

```
MINGW64:/c/Users/Aishwarya Wagh/Assignment1/Practice_Commands
Aishwarya Wagh@DESKTOP-1P4H8RI MINGW64 ~/Assignment1/Practice_Commands (master)
$ git branch
  developer
* master
Aishwarya Wagh@DESKTOP-1P4H8RI MINGW64 ~/Assignment1/Practice_Commands (master)
$ git checkout developer
Switched to branch 'developer'
Aishwarya Wagh@DESKTOP-1P4H8RI MINGW64 ~/Assignment1/Practice_Commands (developer)
$ git branch
* developer
  master
Aishwarya Wagh@DESKTOP-1P4H8RI MINGW64 ~/Assignment1/Practice_Commands (developer)
$ |
```

2. Read about the difference between Git and GitHub.

Git is the tool and GitHub is the service for projects that use Git. Git is a version control system that lets you manage and keep track of your source code history. GitHub is a cloud-based hosting service that lets you manage Git repositories. If you

have open-source projects that use Git, then GitHub is designed to help you better manage them.

3. Read about Git Workflow

4. How many types of version control systems are there?

- **Local Version Control System:**
Local version control system maintains track of files within the local system. This approach is very common and simple. This type is also error prone which means the chances of accidentally writing to the wrong file is higher.
- **Centralized Version Control Systems:**
In this approach, all the changes in the files are tracked under the centralized server. The centralized server includes all the information of versioned files, and list of clients that check out files from that central place.
- **Distributed Version Control System:**
Distributed version control systems come into picture to overcome the drawback of centralized version control system. The clients completely clone the repository including its full history. If any server dies, any of the client repositories can be copied on to the server which help restore the server. Every clone is considered as a full backup of all the data.
Example: Git

5. Explain Branching concept in Git.

If we are creating a new feature for our project, there's a reasonable chance that adding it could break your working code. It's better to start with a prototype, which you would want to design roughly in a different branch and see how it works, before you decide whether to add the feature to the repo's master for others to use. If everyone starts programming on top of our master branch, it will cause a lot of confusion. Everyone has different knowledge and experience in the programming language and the project; some people may write faulty code or simply the kind of code we may not want in our project. Using branches allows us to verify contributions and select which to add to the project. This assumes you are the only owner of the repo and want full control of what code is added to it. In real-life projects, there are multiple owners with the rights to merge code in a repo.

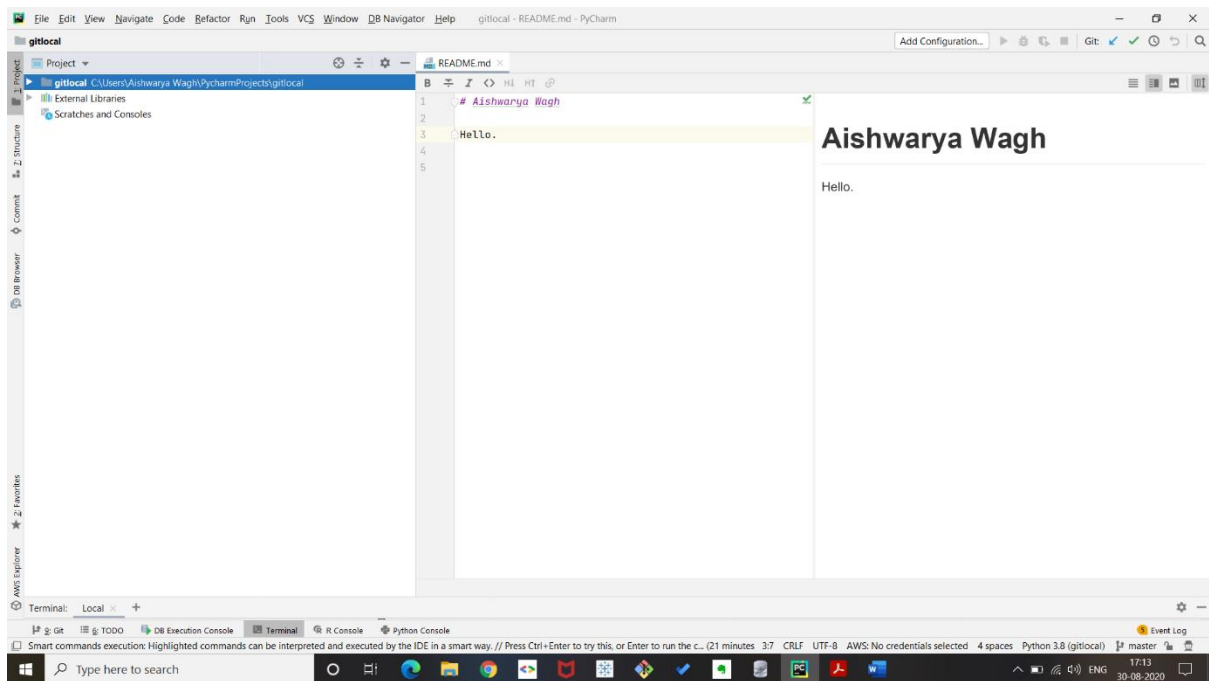
6. Explain Forking Workflow in Git.

Instead, they fork the official repository to create a copy of it on the server. This new copy serves as a public repository no other developers are allowed to push to it, but they can pull changes from it. After they have created their server-side copy, the developer performs a git clone to get a copy of it onto local machine. This serves as their private development environment. When they're ready to publish a local commit, they push the commit to their own public repository not the official one. Then, they file a pull request with the main repository, which lets the project maintainer know that an update is ready to be integrated.

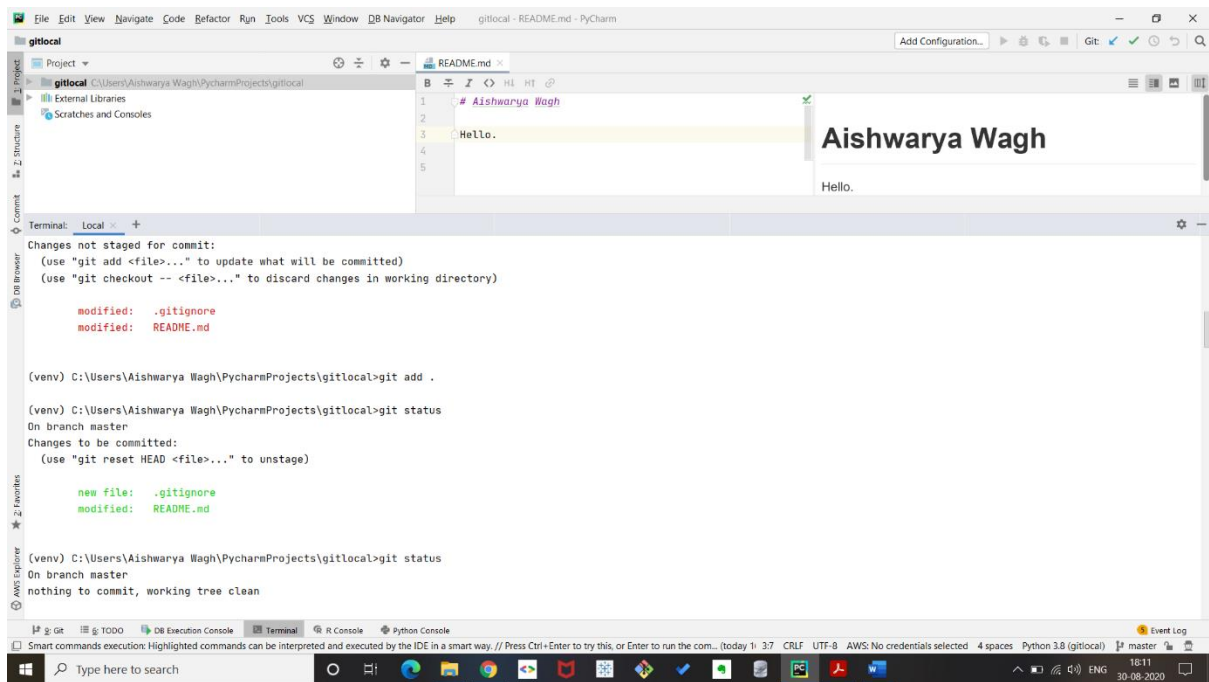
The pull request also serves as a convenient discussion thread if there are issues with the contributed code.

PRACTICAL TASK

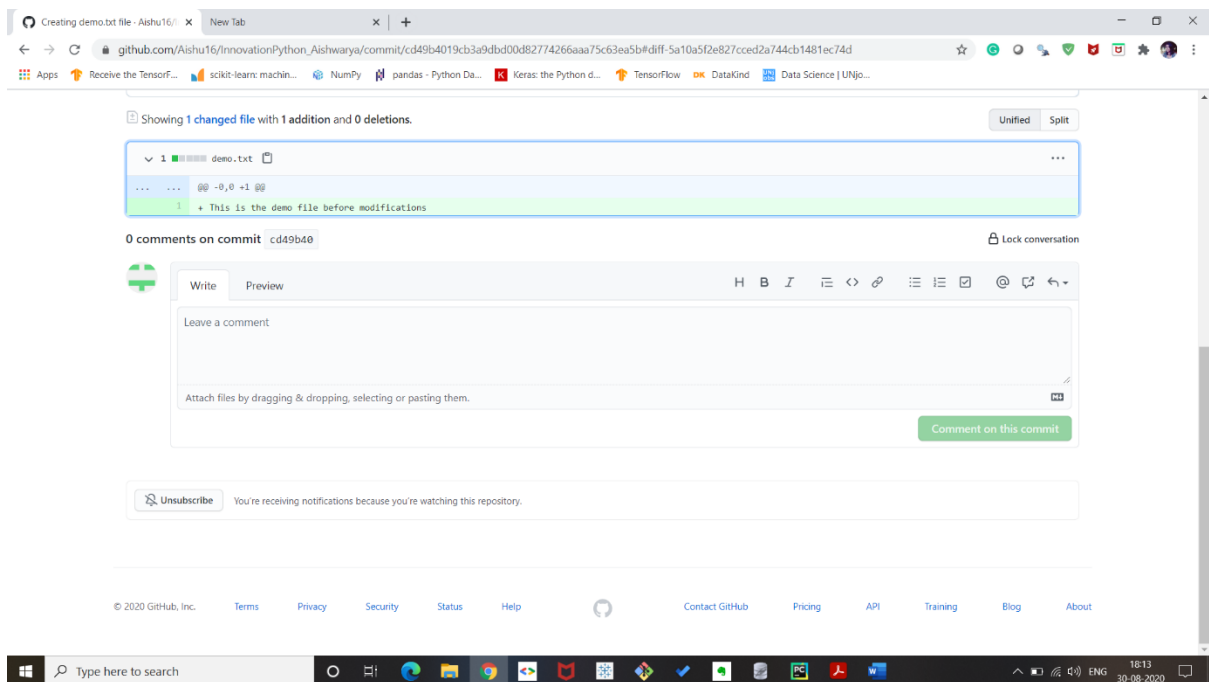
- Initialize an empty Git repository on your local machine with the name “**gitlocal**” and make a **README.md** file in that directory which should contain your name as a heading and a hello message(<https://www.makeareadme.com/>) .



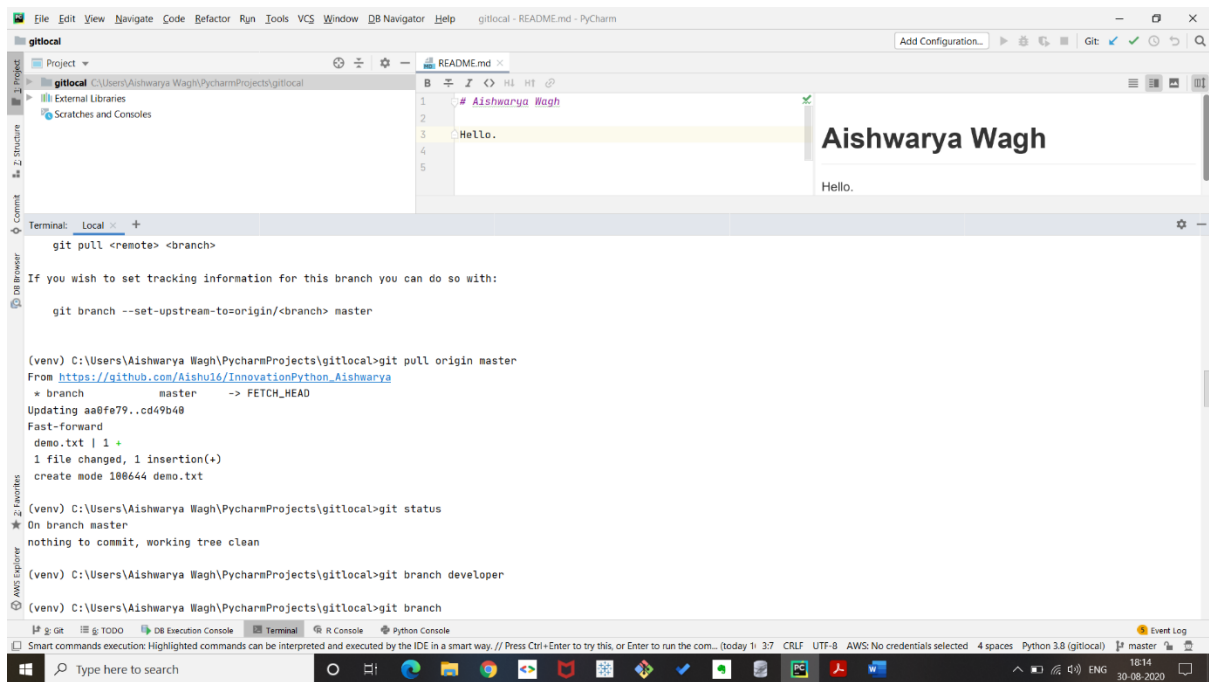
- Now check the status of your git directory and push all the files in that directory to your GitHub repo which you have made in the first step. With a message “First Commit to Git Repo”.



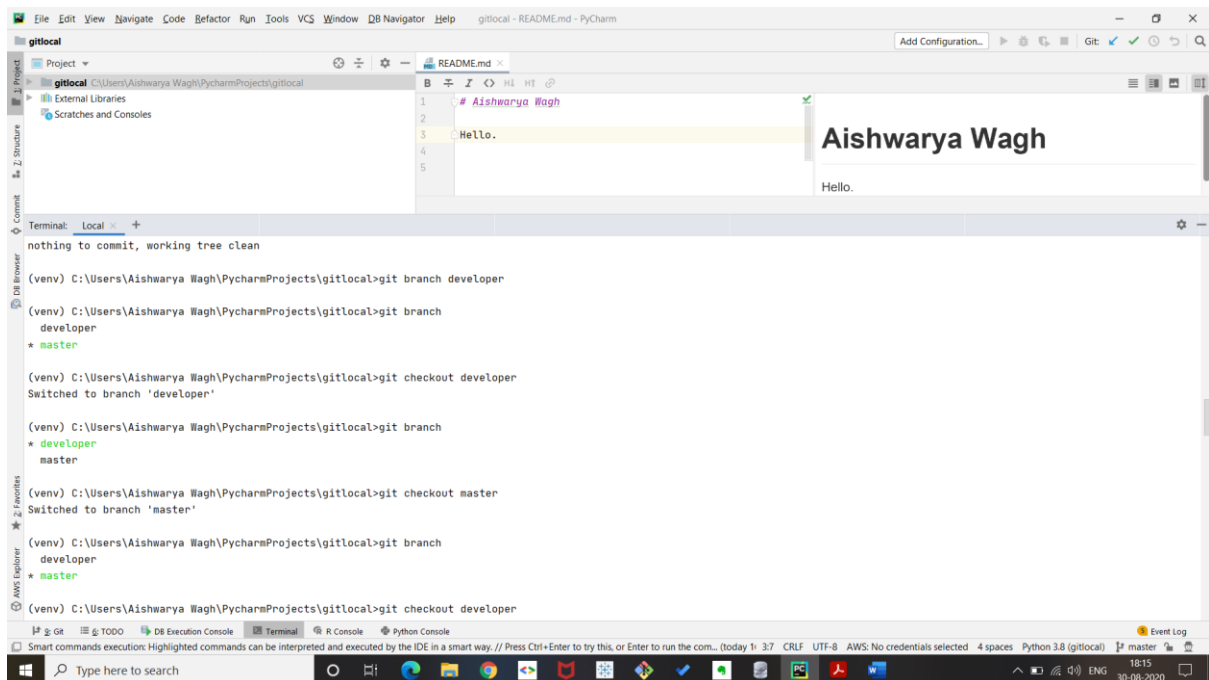
● Now add a file to your Github repo named “demo.txt” from the github console with content : “This is the demo file before modifications”.



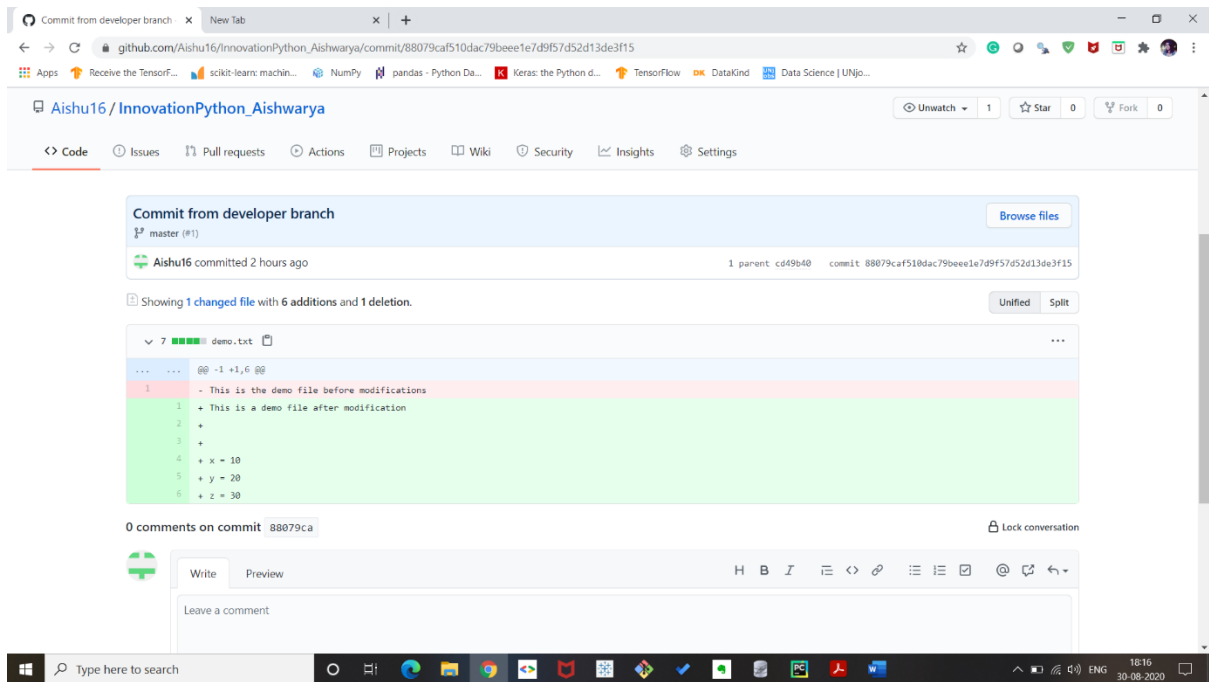
● Pull the changes in your git repo to your local machine git directory named “gitlocal” and check the status for the modifications done in that repo. This time demo should be visible in your local machine.



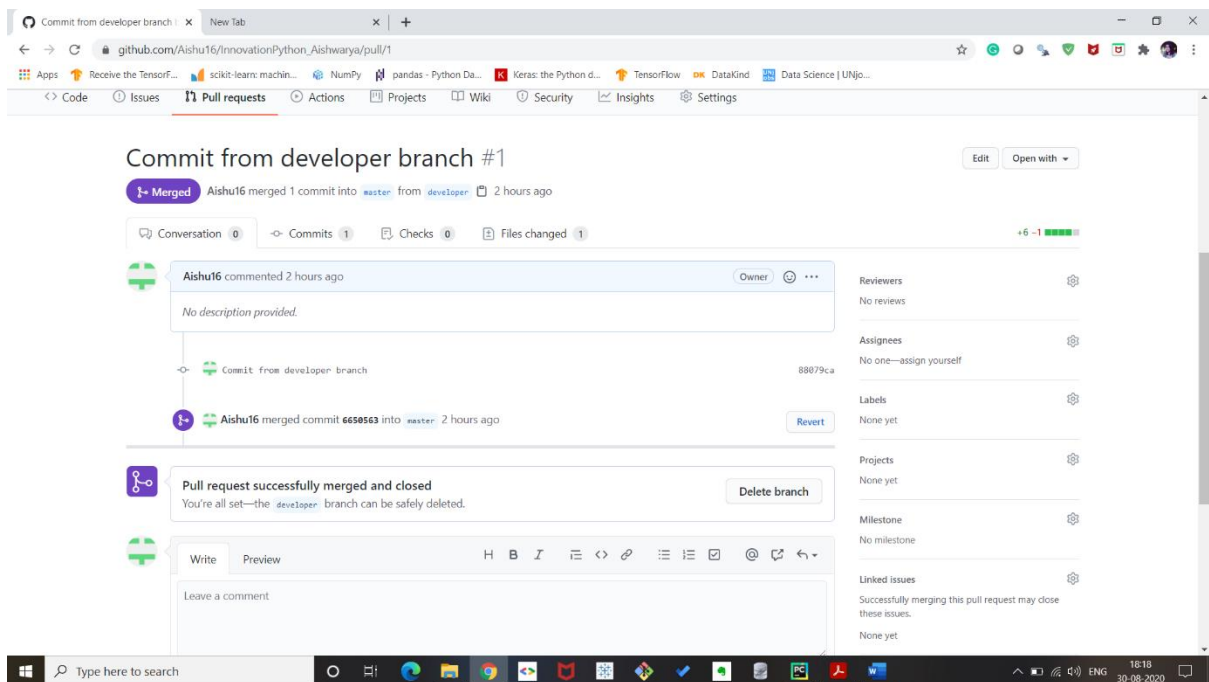
● Now make a new branch in your local machine with the name “**developer**”.



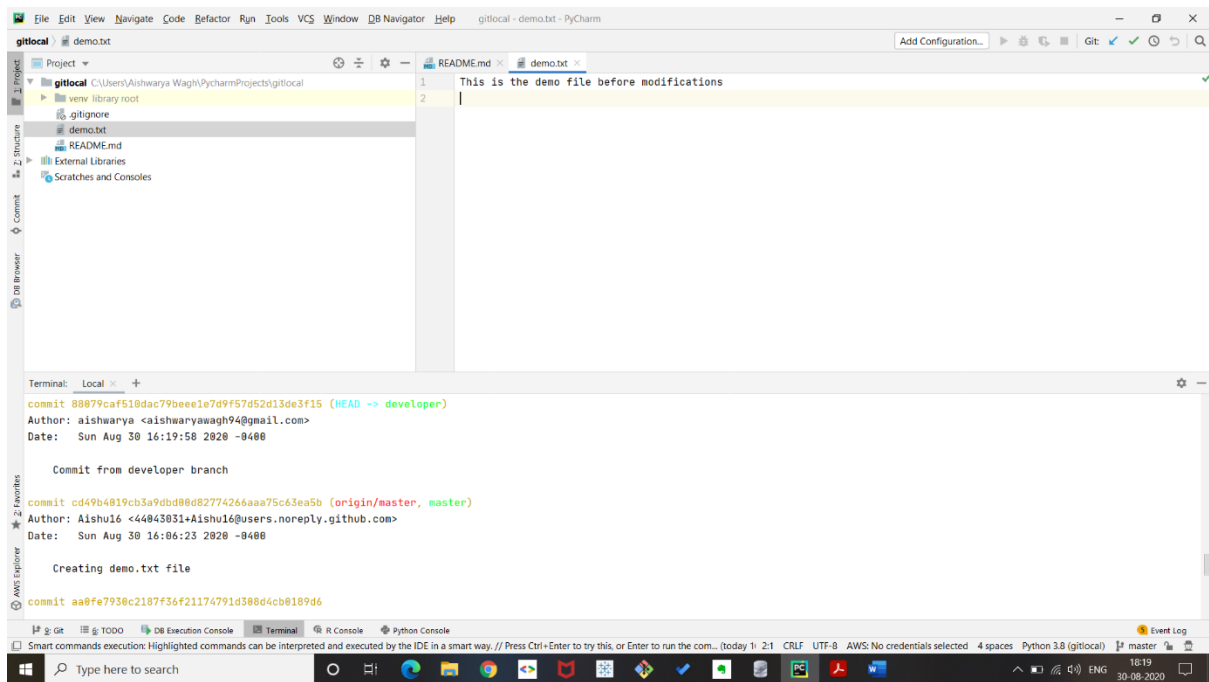
● Edit that demo file and write some content in that eg: “This is a demo file after modification” and push the modifications to your GitHub repo from the **developer** branch with a commit message “Commit from developer branch”.



● Go to the GitHub console and generate a merge request to master branch after checking the modifications.



● After merging you could see the modified content in the demo file. Now revert back to the previous version from terminal. After switching back to the previous version your demo file should have the content: This is a demo file before modification.



● At the end delete the developer branch.

