Git Tasks

4. **How many types of version control systems are there?**
   There are two main categories of version control systems:
   - **Centralized**
   - **Decentralized** (distributed).

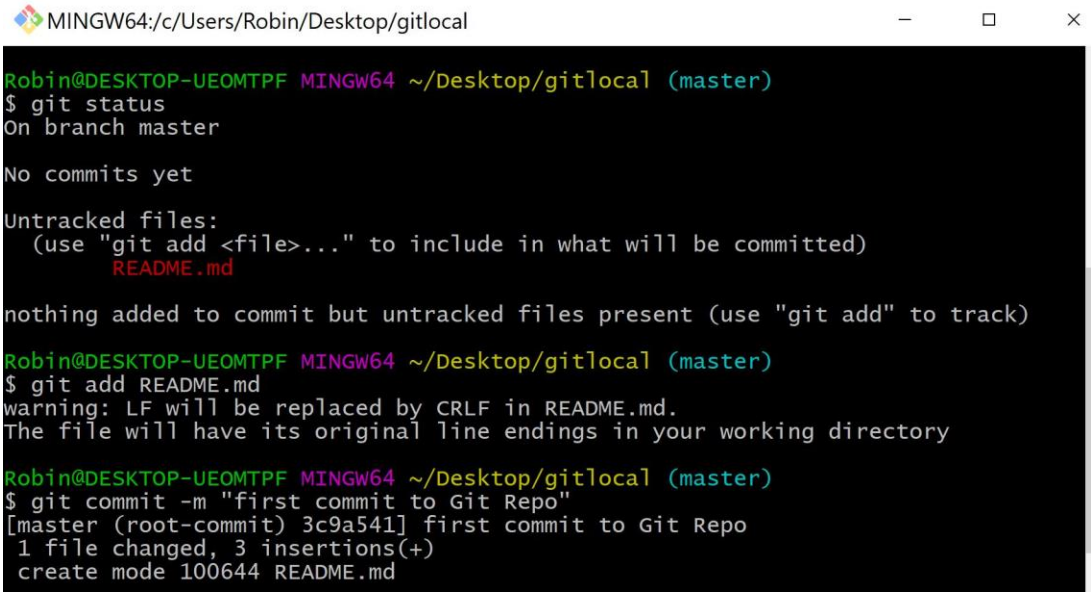
5. **Explain Branching concept in Git.**
   A Git branch is essentially an independent line of development. One can take advantage of branching when working on new features or bug fixes because it isolates his work from that of other team member. Different branches can be merged into any one branch as long as they belong to the same repository. Branching enables one to isolate his work from others. Changes in the primary branch or other branches will not affect his branch, unless he decide to pull the latest changes from those branches.

6. **Explain Forking Workflow in Git.**
   Forking Workflow helps a maintainer of a project open up the repository to contributions from any developer without having to manually manage authorization settings for each individual contributor. This gives the maintainer more of a "pull" style workflow. Most commonly used in open-source projects, the Forking Workflow can also be applied to private business workflows to give more authoritative control over what is merged into a release. This can be useful in teams that have Deploy Managers or strict release cycles.

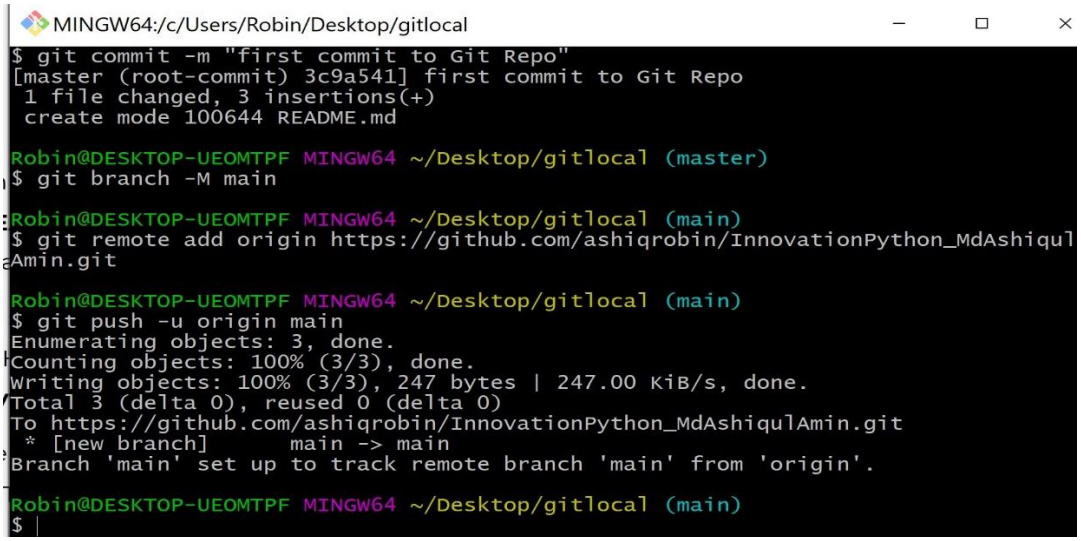
Practical Tasks

- 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 from that directory to your GitHub repo which you have made in the first step (**"InnovationPython_yourname"**). With a massage "First Commit to Git Repo".

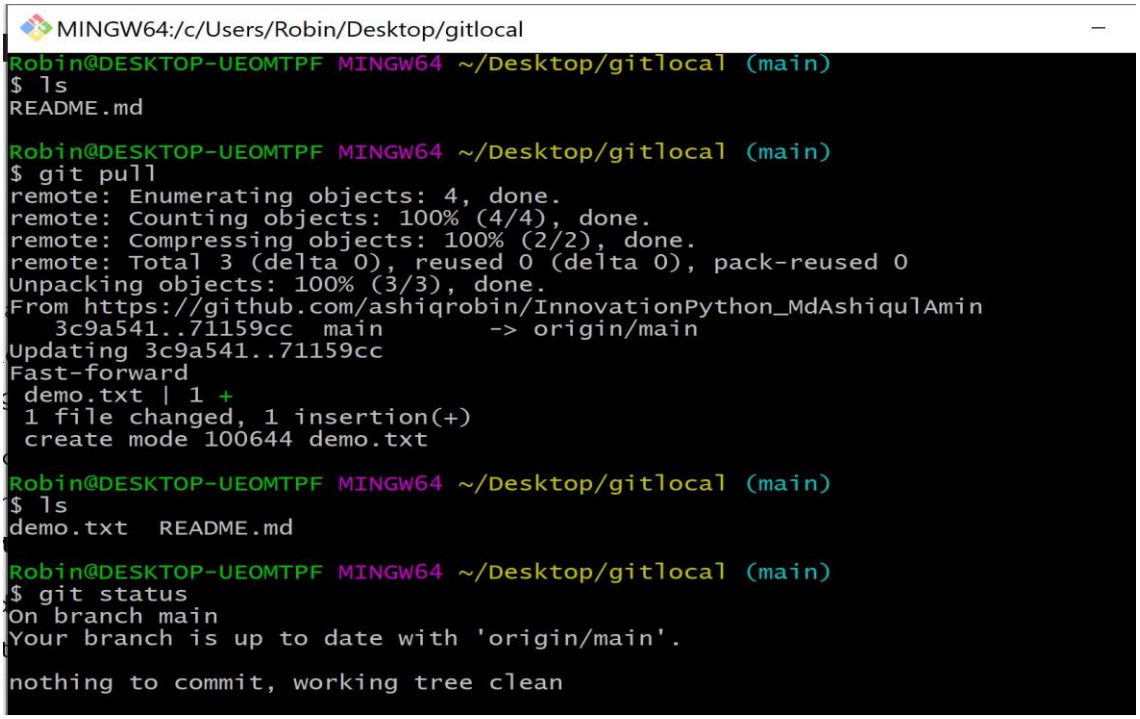


Push the file



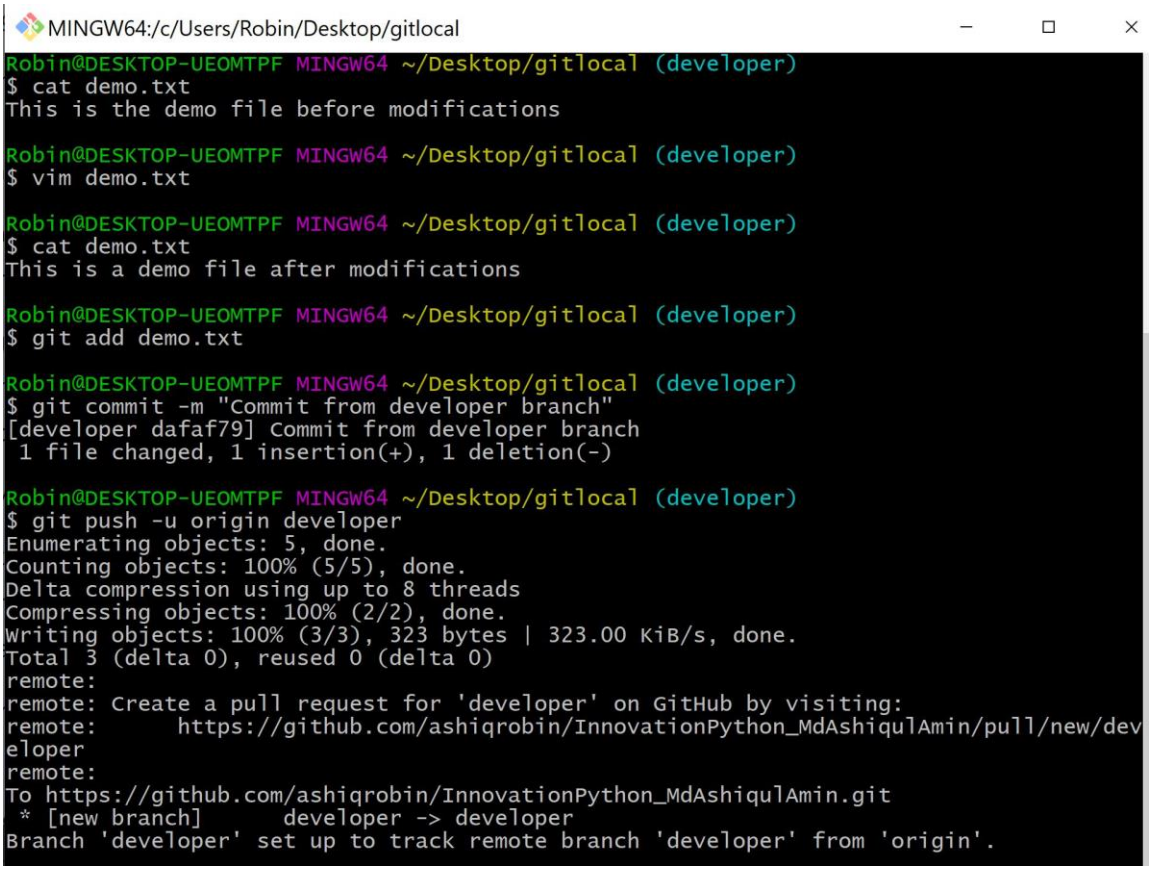
- Now add a file to your Github repo named “demo.txt” from the github console with content : “This is the demo file before modifications”.

- 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 the demo file should be visible in your local machine.

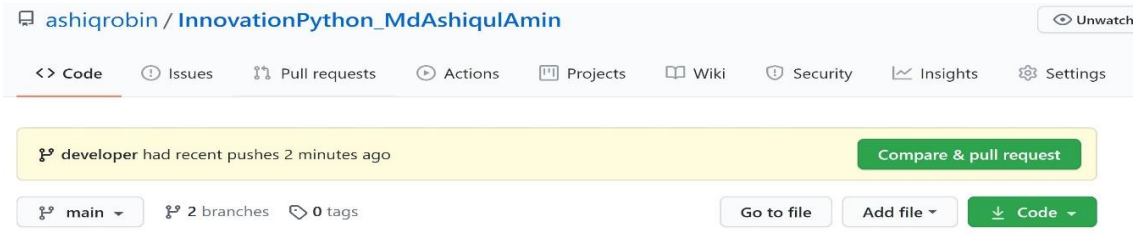


- Now make a new branch in your local machine with the name “ **developer** ”.

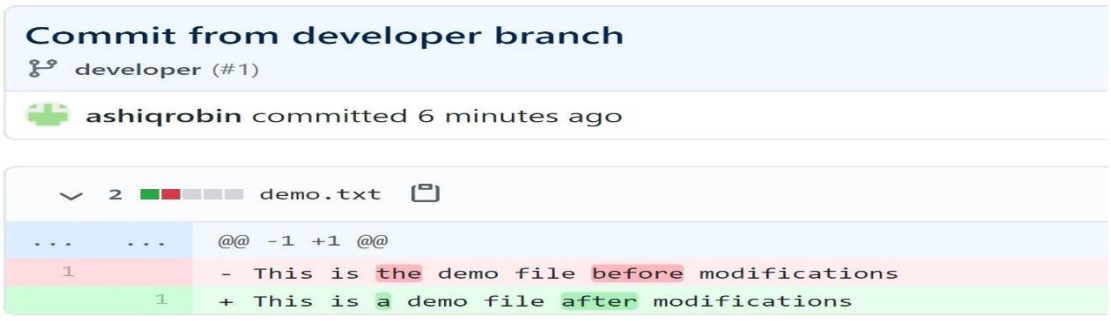
- Edit that demo file and write some content in that e.g. “This is a demo file after modification” and push the modifications to your GitHub repo from the **developer** branch with a commit message “Commit from developer branch”.



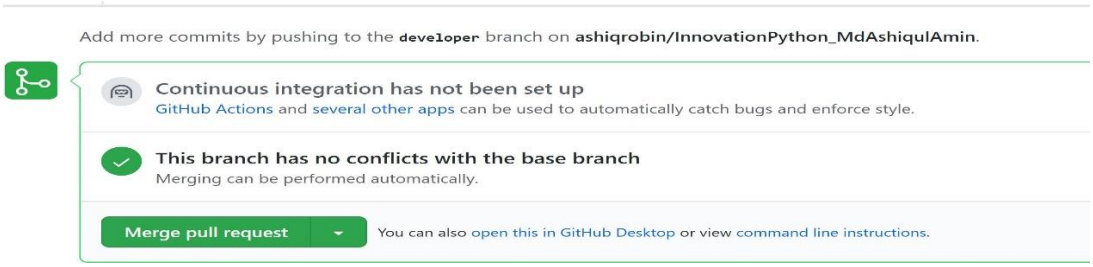
- Go to the GitHub console and generate a merge request to master branch after checking the modifications.



Modifications



- After merging you could see the modified content in the demo file. Now revert back to the previous version from terminal.



modifications

- 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.