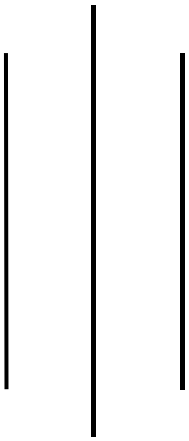
PURBANCHAL UNIVERSITY

KHWOPA ENGINEERING COLLEGE
LIBALI-08, BHAKTAPUR



LAB REPORT ON .NET
LAB NO. 01

SUBMITTED BY:
Name: Piyush Gwayamaru
Roll No. : 770325

SUBMITTED TO:
Department of
Computer Engineering

Submission: 2081/12/09

LAB-1
Git and Github

Theory

Git
Git is distributed version control system used for source code management. It is a free and open-source version control system used to handle small to very large projects efficiently. Git is used to tracking changes in the source code, enabling multiple developers to work together on non-linear development. Linus Torvalds created Git in 2005 for the development of the Linux kernel.

Working with Git

- Initialize Git on a folder, making it a **Repository**
- Git now creates a hidden folder to keep track of changes in that folder
- When a file is changed, added or deleted, it is considered **modified**
- You select the modified files you want to **Stage**
- The **Staged** files are **Committed**, which prompts Git to store a **permanent** snapshot of the files
- Git allows you to see the full history of every commit.
- You can revert back to any previous commit.
- Git does not store a separate copy of every file in every commit, but keeps track of changes made in each commit!

GitHub
GitHub is a web-based platform that provides version control and collaboration tools for developers, enabling them to manage, store, and track changes in their code efficiently. Built on Git, an open-source version control system created by Linus Torvalds, GitHub allows developers to work on projects individually or as a team. It offers features such as repositories for storing code, branches for parallel development, pull requests for code reviews, and issue tracking for project management.

Forking & Cloning

- **Forking** creates a personal copy of another user's repository.
- **Cloning** downloads a repository to a local computer for offline development.

Common Git and Github Commands

Git Configuration
git config --global user.name "Your Name"
This command sets the global username for the Git commits.
git config --global user.email "your_email@example.com"
This command sets the global email associated with Git commits.

Git Initializing
git init
This command initialize a new git repository in the current directory.

Git Linking
git remote add origin <repo>
This command links new repository of github with the local codes.

Git Staging and Commits
git add .
This command add files to the staging area.
git commit -m "message"
This command commit and save changes of stage area with a message.

Git Status and Log
git status
This command check the status of the working directory.
git log
This command view commit history.

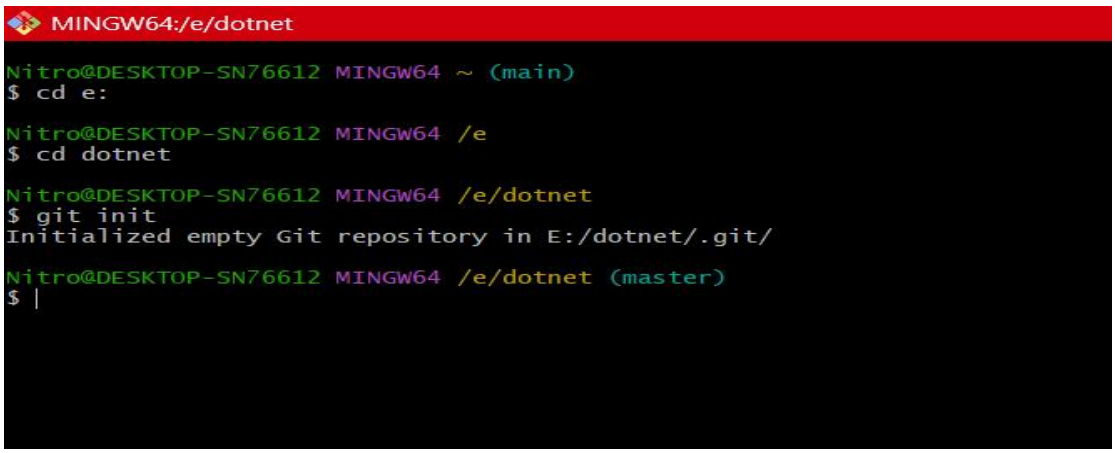
Git Branching and Merging
git branch
This command list all the branches exist in the repository.
git branch <branch_name>
This command creates new branch for separate development.
git checkout <branch>
This command switches to another branch.
git switch <branch_name>
This command switches to another branch.
git merge <branch_name>
This command merges a specified branch into the current branch.

Git Push and Pull
git push -u origin <branch_name>
This command uploads commits to a remote repository
git pull origin
This command fetch and merge changes from a remote repository

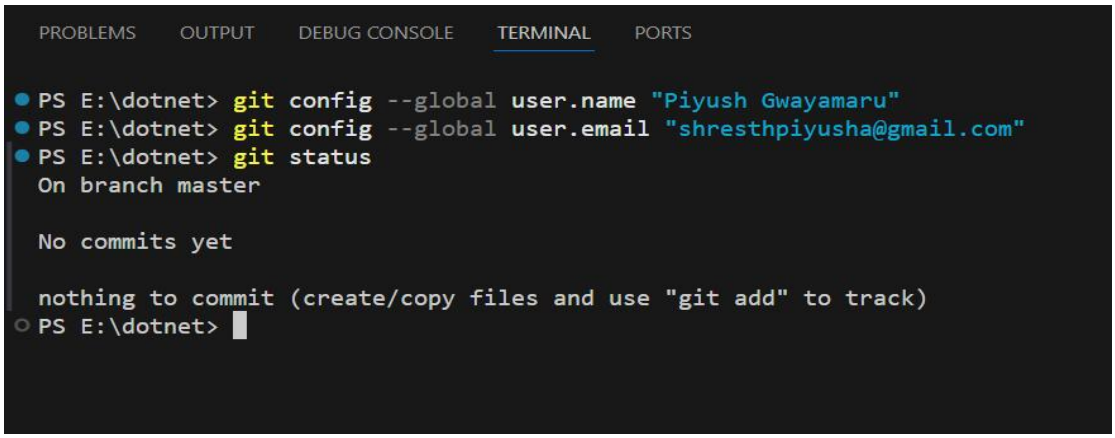
Git Clone
git clone <repo_url>
This command copy(clone) an existing repository.

Lab Work

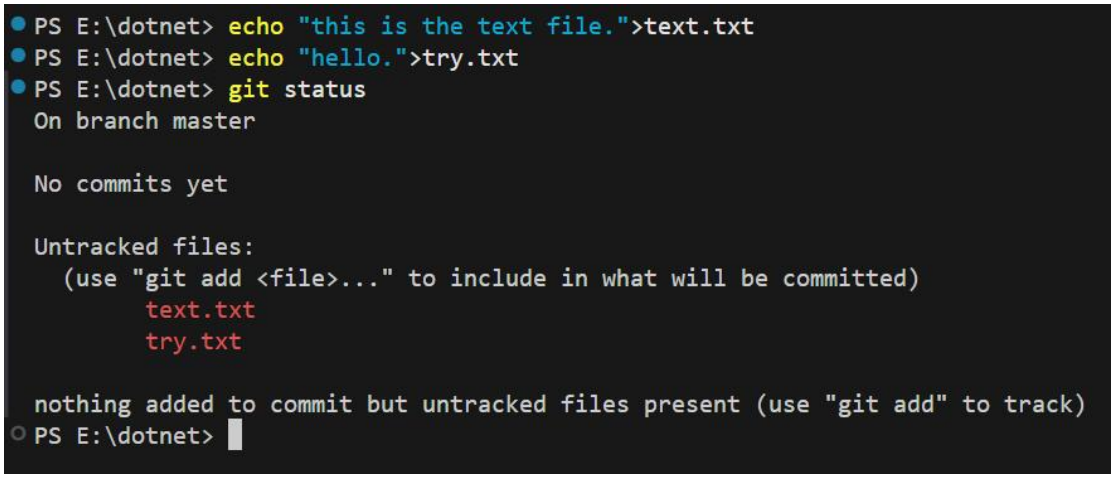
Initially, a folder is created to create, change the files using the version control git with different commands and git is initialized.



We set the global username and email of the GitHub. As no any files are created there is nothing to commit.



Then we create two empty files in master branch and with the help of echo we insert text on those files and checked the status , it is in untracked stage and we sent the files to the staging stage.



The files are then added for staging and commit the files with the message such that the files are stored in the local repository. Then we check git status, there was nothing left to commit.

```
PS E:\dotnet> git add .
PS E:\dotnet> git commit -m "initial commit"
[master (root-commit) d3b530d] initial commit
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 text.txt
 create mode 100644 try.txt
PS E:\dotnet> git status
On branch master
nothing to commit, working tree clean
PS E:\dotnet> git log
commit d3b530d2aaa3661077e0a6a0daefe63b84c7aa1e (HEAD -> master)
Author: Piyush Gwayamaru <shresthpiyusha@gmail.com>
Date:   Fri Mar 21 12:37:19 2025 +0545

    initial commit
PS E:\dotnet>
```

We again made certain changes in file text.txt to see certain changes in the file status. And again commit it so that all of the files are saved in the local repository.

```
PS E:\dotnet> 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:   text.txt

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

We then add the files in the remote repository by creating the repository in the GitHub and copying the url of the repo and using the above code.

```
PS E:\dotnet> git remote add origin https://github.com/piyushgwayamaru/dotNet.git
```

After that we push the files to the created repository.

```
PS E:\dotnet> git push -u origin master
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 12 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (7/7), 680 bytes | 680.00 KiB/s, done.
Total 7 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/piyushgwayamaru/dotNet.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```

We checked the existing branch in our local repository. Then we create branches for working different version of programs without affecting the main code.

```
PS E:\dotnet> git branch
  * master
PS E:\dotnet> git branch developer
PS E:\dotnet> git branch
    developer
  * master
```

We switched to new branch “developer” branch where we modify and add different files without affecting the main code. We here added new files “calculation.py”. Initially it is in untracked stage.

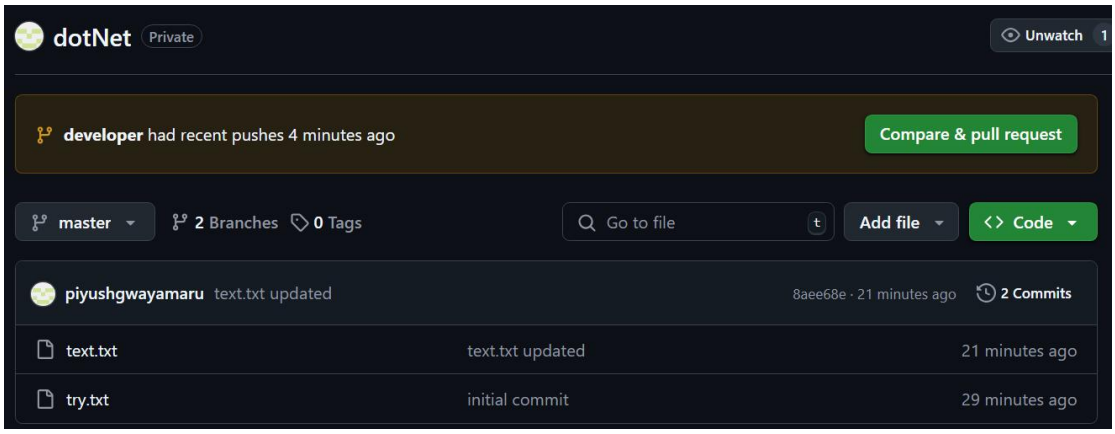
```
PS E:\dotnet> git checkout developer
Switched to branch 'developer'
PS E:\dotnet> git status
On branch developer
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        calculation.py

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

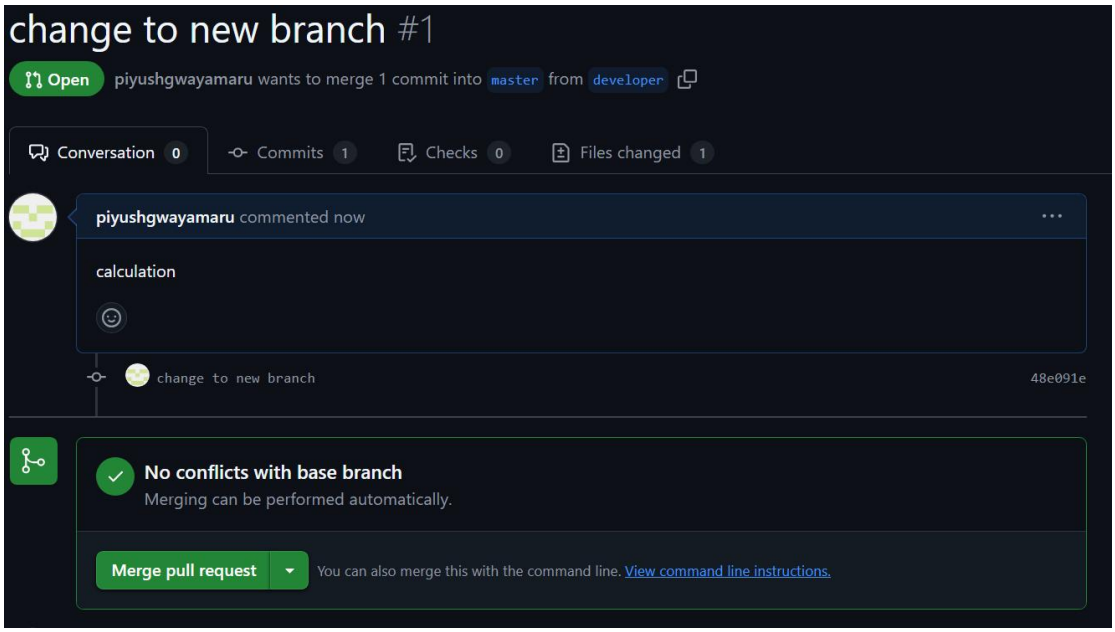
Then we commit the changes and push the branch in the GitHub to make sure the branch is visible to other users of the repository.

```
PS E:\dotnet> git add .
PS E:\dotnet> git commit -m "change to new branch"
[developer 48e091e] change to new branch
 1 file changed, 10 insertions(+)
 create mode 100644 calculation.py
PS E:\dotnet> git push -u origin developer
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 489 bytes | 489.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'developer' on GitHub by visiting:
remote:      https://github.com/piyushgwayamaru/dotNet/pull/new/developer
remote:
To https://github.com/piyushgwayamaru/dotNet.git
 * [new branch]      developer -> developer
branch 'developer' set up to track 'origin/developer'.
```

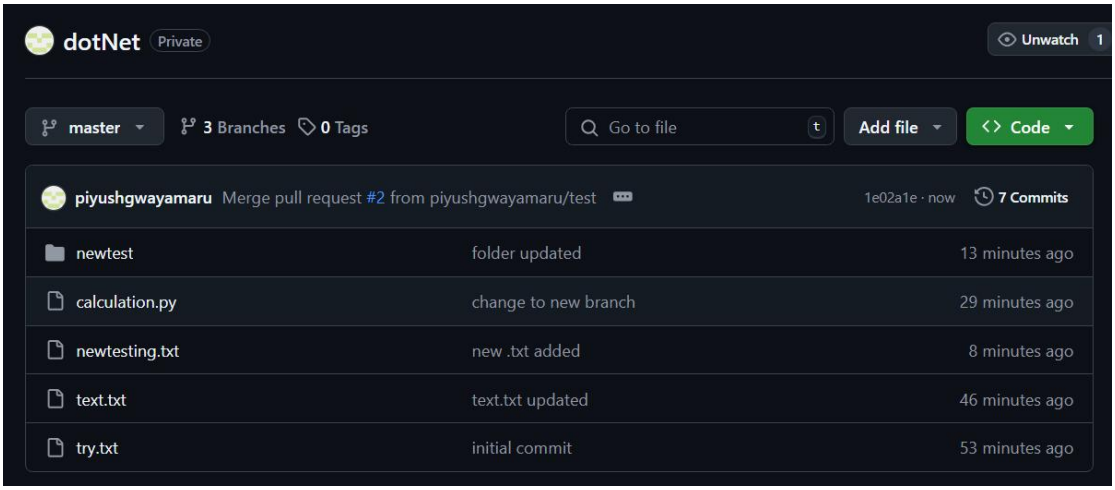
Then we create pull request for merging the latest branch pushed.



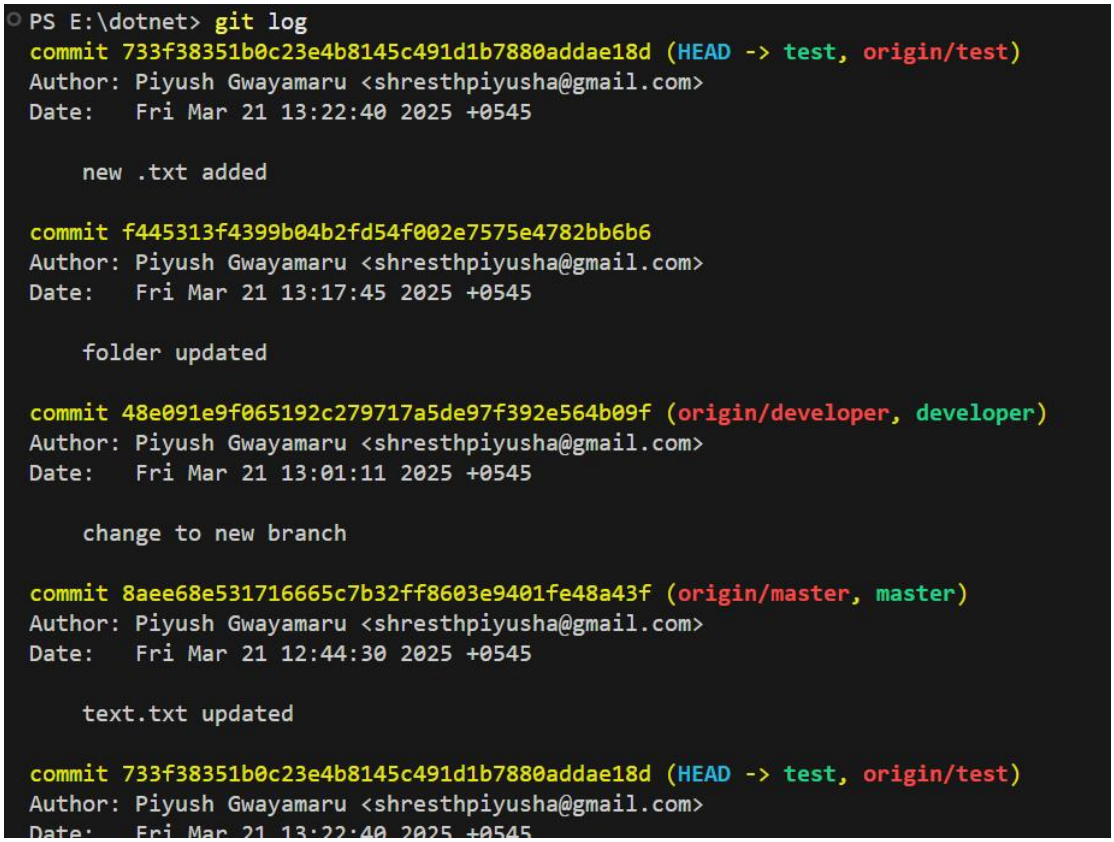
There was no any conflict so we merged the new branch into master branch.



After completion of merge we can see all the files in master branch.



With the help of command “git log” we viewed the history of the commits.



Conclusion

In this lab, we learned and performed different commands for initializing, linking, commiting, branching and merging in Git and Github.