PURBANCHAL UNIVERSITY

KHWOPA ENGINEERING COLLEGE

LIBALI-08, BHAKTAPUR

LAB REPORT ON **.NET**

LAB NO. 01

<u>SUBMITTED BY:</u>	**<u>SUBMITTED TO:</u>**
Name: Sumina Awa	Department of Computer Engineering
Roll No.: 770345	Khwopa Engineering College
Group: ‘B’	Libali-08, Bhaktapur

Submission: 2081/12/09

Theory:

1. **Git:**

   Git is a **distributed version control system (DVCS)** that helps developers track changes in their code, collaborate with others, and manage different versions of a project efficiently. It was created by **Linus Torvalds** in 2005 for Linux kernel development. It allows multiple developers to collaborate efficiently by managing different version of project. Git enables branching, merging and reverting changes, making code management easier. It is widely used open-source and commercial projects. Popular platform like GitHub, GitLab, and Bitbucket provide remote repositories for Git-based collaboration.

   Git Workflow

   1. Working Directory – The files you are currently working on.
   2. Staging Area (Index) – Files that are marked to be committed.
   3. Repository (Local Repo) – The committed files stored locally.
   4. Remote Repository – A shared repository (e.g., GitHub, GitLab, Bitbucket).

   Why Use Git?

   - Version Control: Tracks changes in files over time.
   - Collaboration: Multiple developers work on the same project simultaneously.
   - Branching & Merging: Work on new features without affecting the main project.
   - Backup & Recovery: Keeps a history of changes, preventing data loss.
   - Speed & Efficiency: Git is lightweight, fast compared to other version control systems like SVN.

2. **GitHub**

   GitHub is a **web-based platform** for version control and collaboration using Git. It allows developers to store, manage, and share code repositories efficiently. GitHub supports features like branching, pull requests, issue tracking, and CI/CD integration. It is widely used for open-source and private projects, enabling seamless teamwork. GitHub also provides cloud-based hosting, making it accessible from anywhere.

Forking & Cloning

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

General Git and GitHub Commands:

Category	Command	Description
Git Configuration	`git config --global user.name “Your Name”`	Sets the global username for Git commits.
	`git config --global user.email “your_email@example.com”`	Sets the global email associated with Git commits.
Initializing	`git init`	Initializes a new Git repository in the current directory.
Staging	`git add .`	Stages all changes and new files for commit.
Commits	`git commit -m “Your commit message”`	Saves the staged changes with a descriptive message.
Branching and Merging	`git branch`	Lists all the branches in the repository.
	`git branch <branch_name>`	Creates a new branch for separate development.
	`git checkout <branch_name>` / `git switch <branch_name>`	Switches to the specified branch.
	`git merge <branch_name>`	Merges changes from the specified branch into the current branch.
Pushing	`git push -u origin <branch_name>`	Uploads the local changes to the remote repository.
Pulling	`git pull origin <branch_name>`	Fetches and merges the latest changes from the remote repository.
Status	`git status`	Shows the current state of files (modified, staged, or untracked).
Logs	`git log`	Displays the commit history of the repository.
Remote Repository	`git remote add origin <repo_url>`	Links the local repository to a remote repository on GitHub.

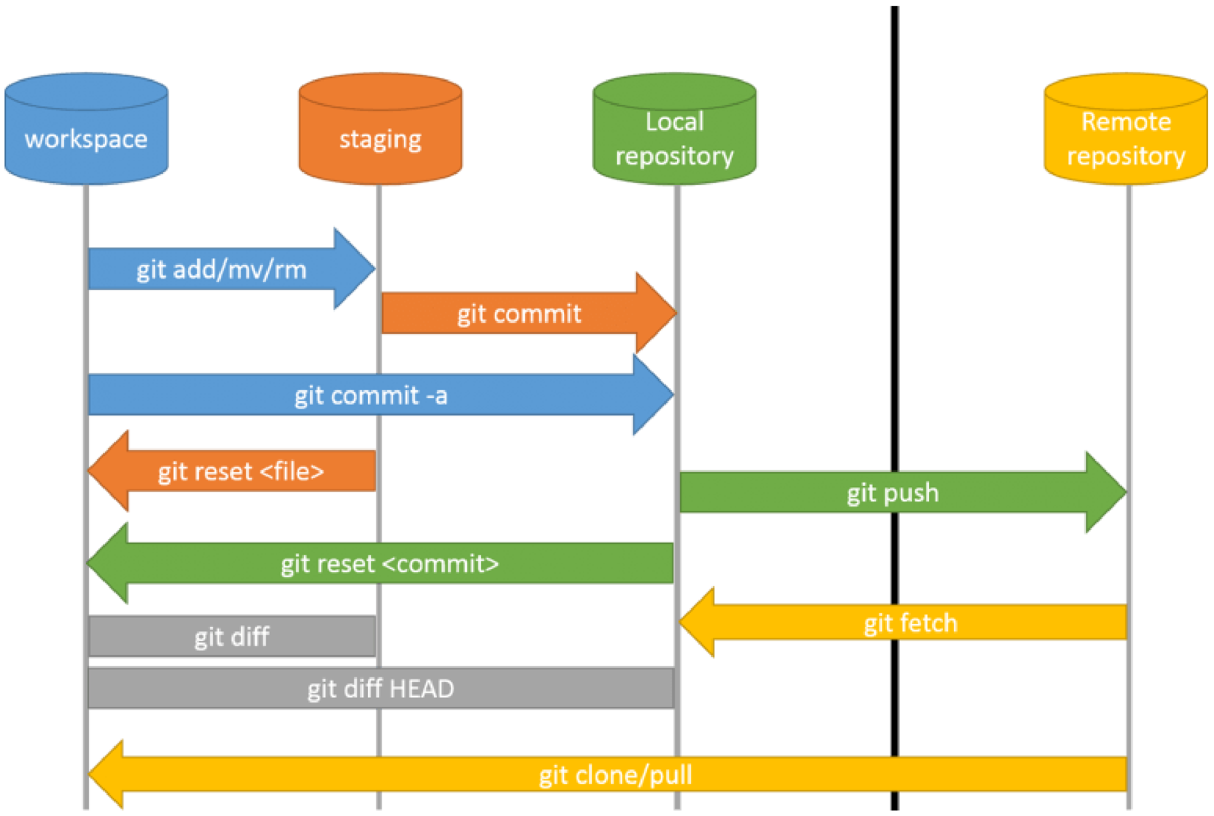
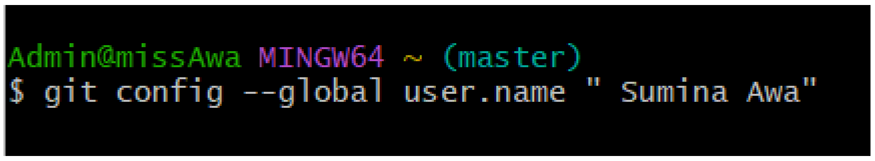


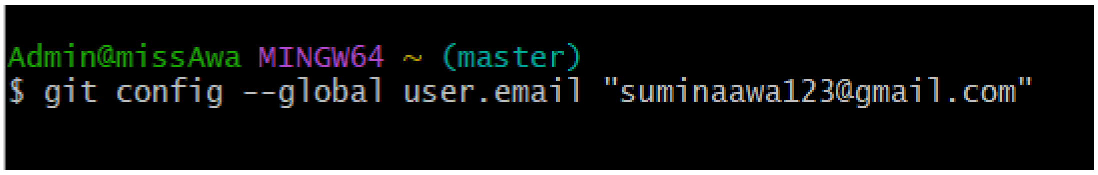
Fig: Git Workflows

Lab Works

1. First set the global username and email of the GitHub.

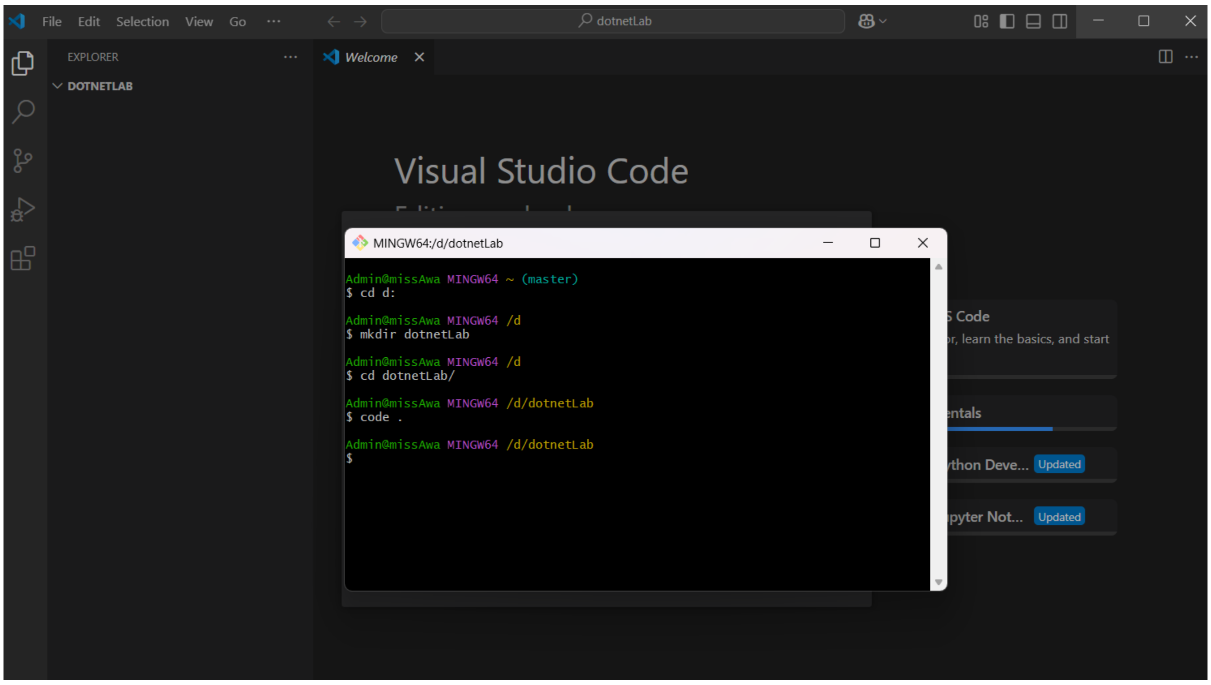


```
Admin@missAwa MINGW64 ~ (master)
$ git config --global user.name " Sumina Awa"
```



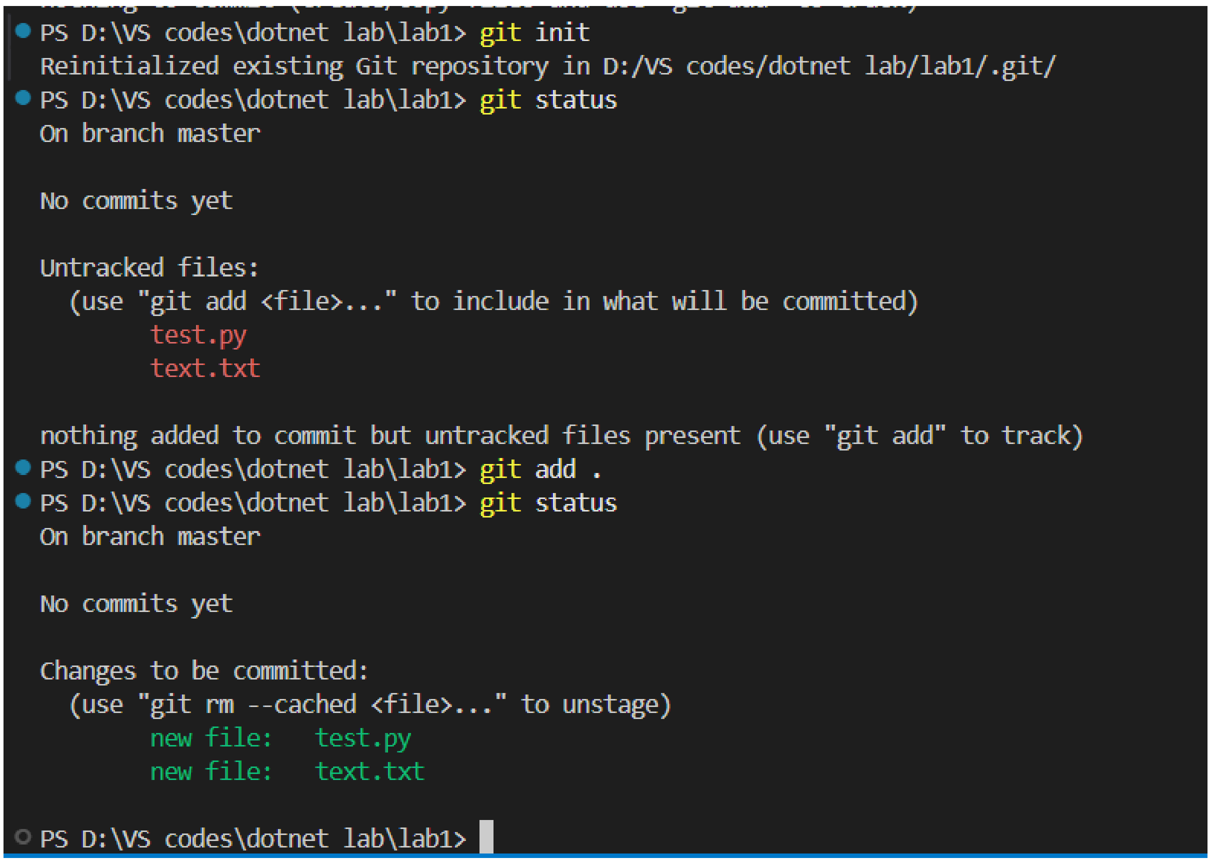
```
Admin@missAwa MINGW64 ~ (master)
$ git config --global user.email "suminaawa123@gmail.com"
```

2. Create a folder and inside it files as per the user desire so that we can identify the changes inside the file using the version control (Git).



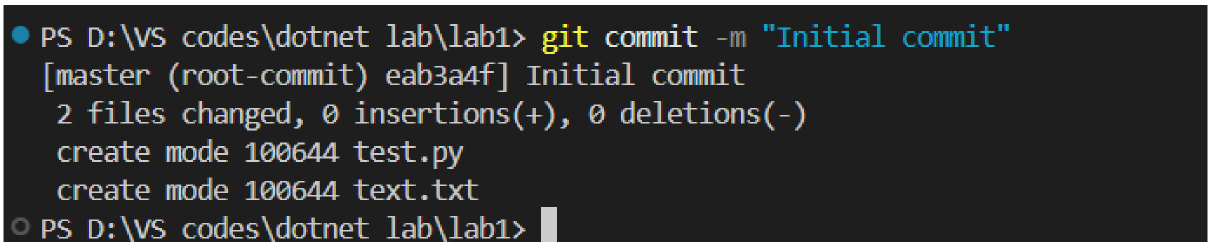
```
Admin@missAwa MINGW64 ~ (master)
$ cd d:

Admin@missAwa MINGW64 /d
$ mkdir dotnetLab

Admin@missAwa MINGW64 /d
$ cd dotnetLab/

Admin@missAwa MINGW64 /d/dotnetLab
$ code .

Admin@missAwa MINGW64 /d/dotnetLab
$
```

3. On creating the new files, initially the files are in the untracked stage so sent the untracked files to the staging stage. To do so first initialize the directory and staged the files.



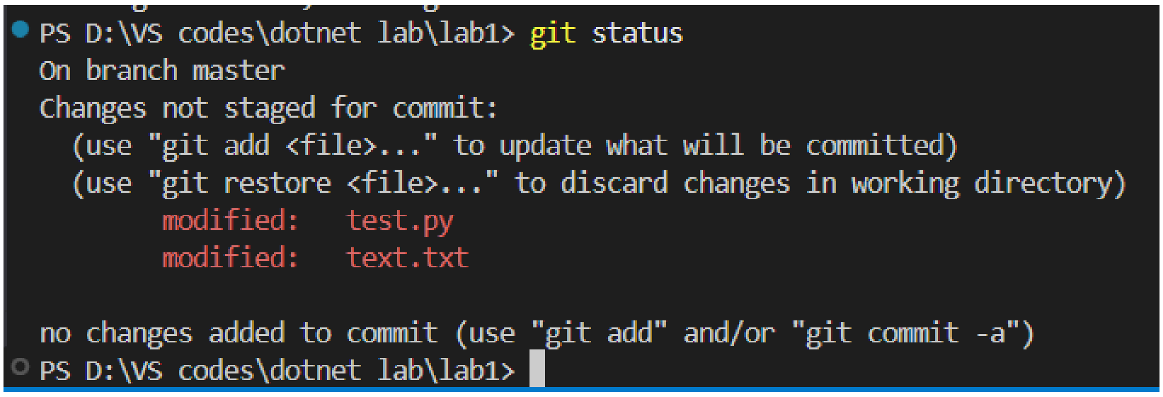
```
PS D:\VS codes\dotnet lab\lab1> git init
Reinitialized existing Git repository in D:/VS codes/dotnet lab/lab1/.git/
PS D:\VS codes\dotnet lab\lab1> git status
On branch master

No commits yet

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

nothing added to commit but untracked files present (use "git add" to track)
PS D:\VS codes\dotnet lab\lab1> git add .
PS D:\VS codes\dotnet lab\lab1> git status
On branch master

No commits yet

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

PS D:\VS codes\dotnet lab\lab1>
```

4. Now commit the files such that the files are stored in the local repository.



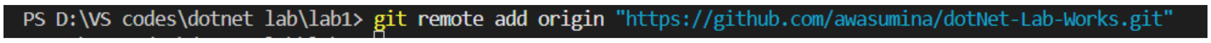
```
PS D:\VS codes\dotnet lab\lab1> git commit -m "Initial commit"
[master (root-commit) eab3a4f] Initial commit
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 test.py
 create mode 100644 text.txt
PS D:\VS codes\dotnet lab\lab1>
```

5. Make certain changes inside the file to see the changes in the file status.



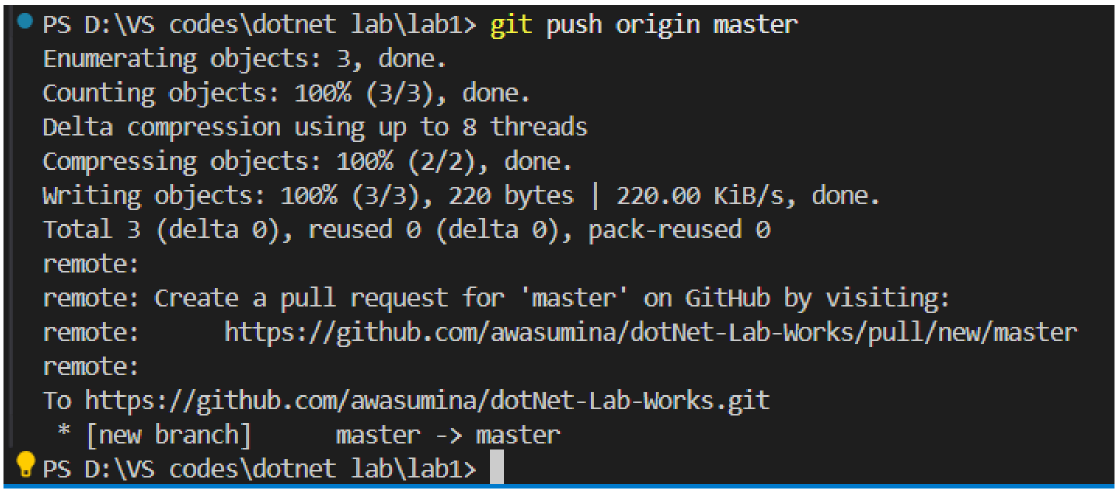
```
PS D:\VS codes\dotnet lab\lab1> git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   test.py
        modified:   text.txt

no changes added to commit (use "git add" and/or "git commit -a")
PS D:\VS codes\dotnet lab\lab1>
```

6. After changing the contents in the file **"test.py"** add the file and commit it. All of these files are saved in the local repository. Now to add these files in the remote repository create the repository in the GitHub and copy the url of the repo and use the following code.



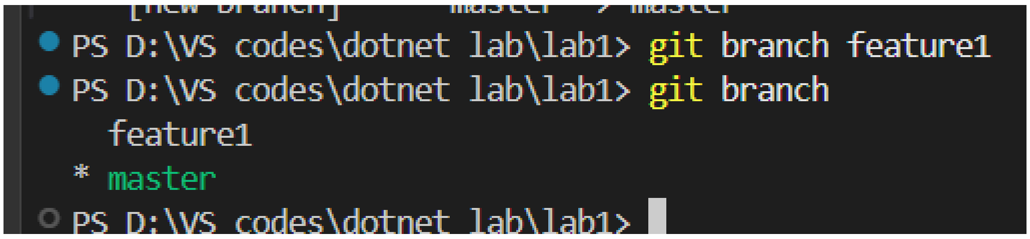
```
PS D:\VS codes\dotnet lab\lab1> git remote add origin "https://github.com/awasumina/dotNet-Lab-Works.git"
```

7. Now push the files in the repository created.



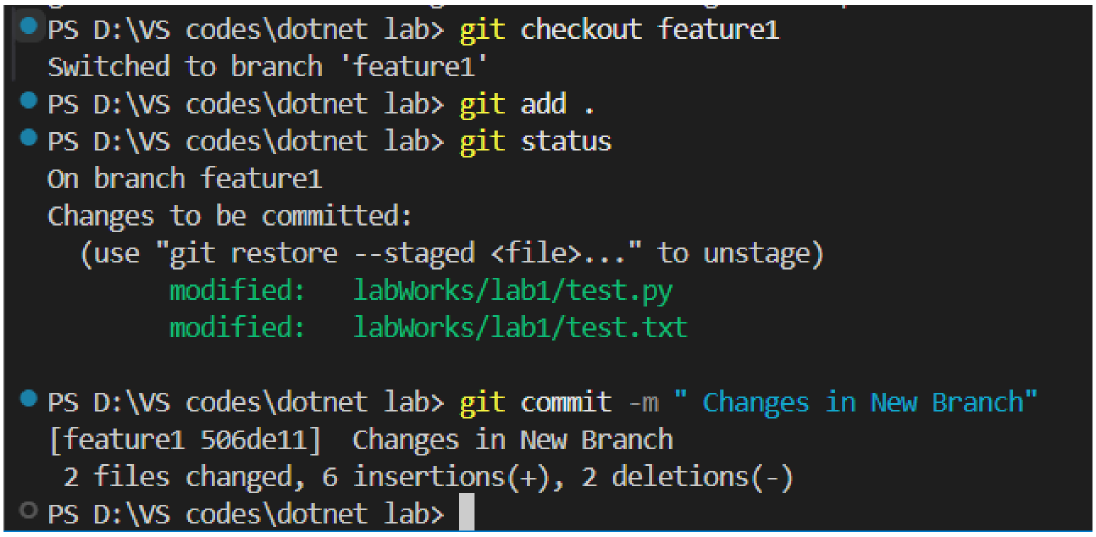
```
PS D:\VS codes\dotnet lab\lab1> git push origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 220 bytes | 220.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:      https://github.com/awasumina/dotNet-Lab-Works/pull/new/master
remote:
To https://github.com/awasumina/dotNet-Lab-Works.git
 * [new branch]      master -> master
PS D:\VS codes\dotnet lab\lab1>
```

8. Now creating branches, allowing the work on different version of a project without affecting the main codebase.



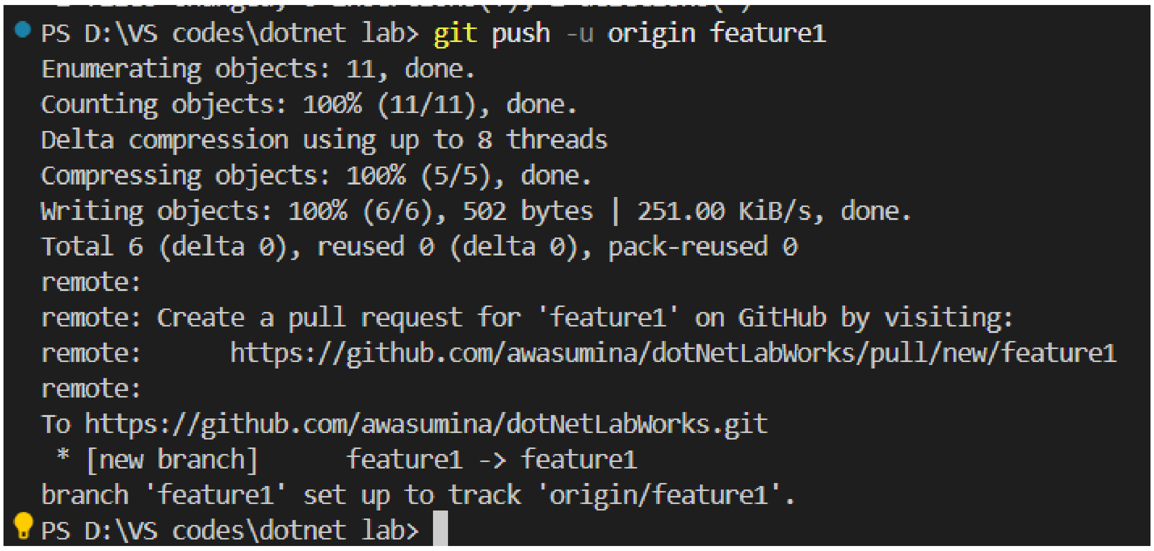
```
PS D:\VS codes\dotnet lab\lab1> git branch feature1
PS D:\VS codes\dotnet lab\lab1> git branch
  feature1
* master
PS D:\VS codes\dotnet lab\lab1>
```

9. Moving on to the recently created branch to modify the contents in the file without affecting the main codebase.



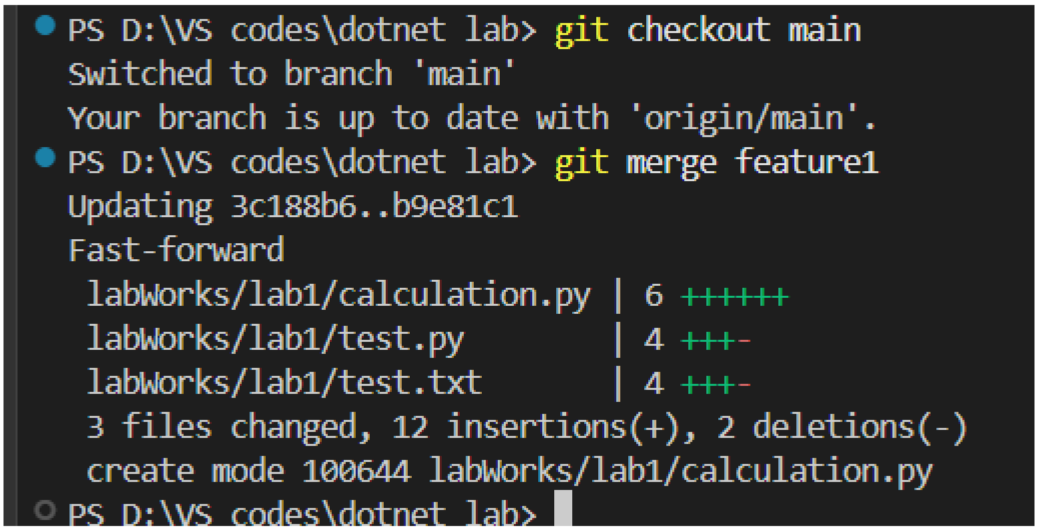
```
PS D:\VS codes\dotnet lab> git checkout feature1
Switched to branch 'feature1'
PS D:\VS codes\dotnet lab> git add .
PS D:\VS codes\dotnet lab> git status
On branch feature1
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   labWorks/lab1/test.py
        modified:   labWorks/lab1/test.txt

PS D:\VS codes\dotnet lab> git commit -m " Changes in New Branch"
[feature1 506de11]  Changes in New Branch
 2 files changed, 6 insertions(+), 2 deletions(-)
PS D:\VS codes\dotnet lab>
```

10. To change the branch, we can use the command *"git switch main"*. To make sure the branch is visible to other users of the repository push the branch in the GitHub.

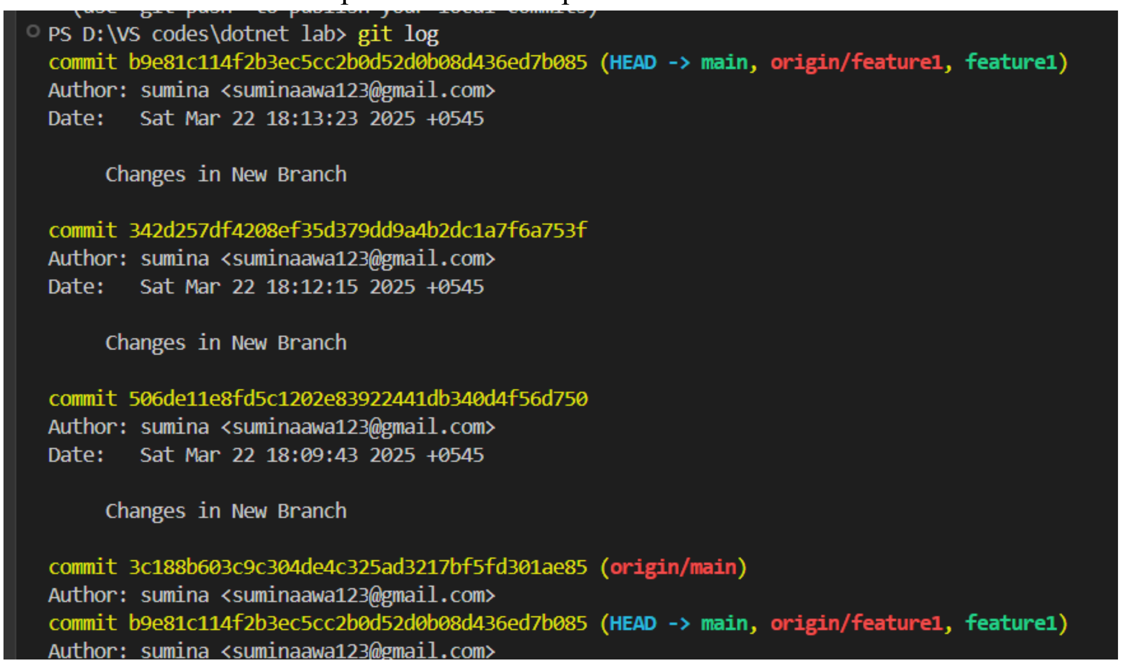


```
PS D:\VS codes\dotnet lab> git push -u origin feature1
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 502 bytes | 251.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'feature1' on GitHub by visiting:
remote:      https://github.com/awasumina/dotNetLabWorks/pull/new/feature1
remote:
To https://github.com/awasumina/dotNetLabWorks.git
 * [new branch]      feature1 -> feature1
branch 'feature1' set up to track 'origin/feature1'.
PS D:\VS codes\dotnet lab>
```

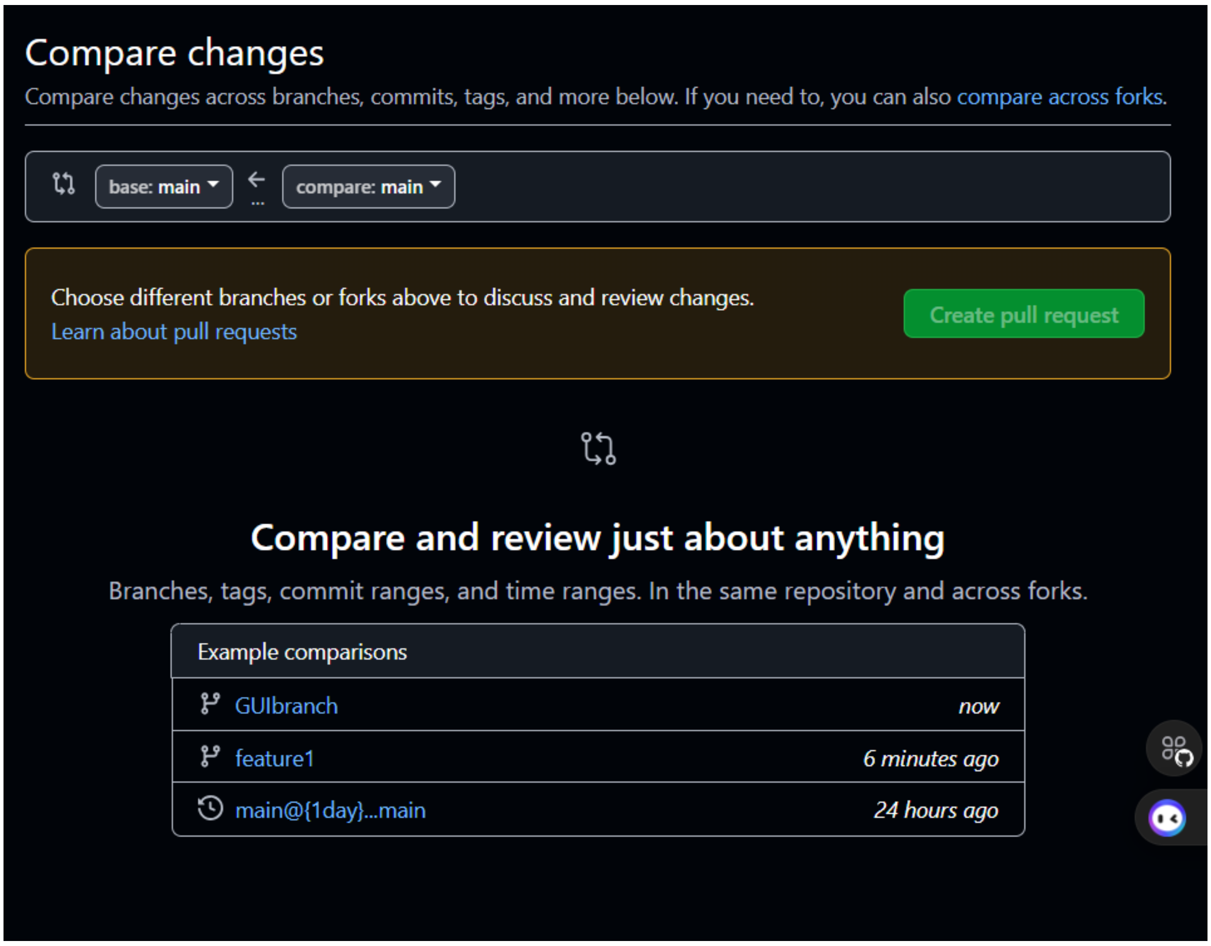
11. Merging the branches such that the changes in the new branch or new features added in the new branch is added to the main code base.


```
PS D:\VS codes\dotnet lab> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS D:\VS codes\dotnet lab> git merge feature1
Updating 3c188b6..b9e81c1
Fast-forward
 labWorks/lab1/calculation.py | 6 ++++++
 labWorks/lab1/test.py        | 4 +++-
 labWorks/lab1/test.txt       | 4 +++-
 3 files changed, 12 insertions(+), 2 deletions(-)
 create mode 100644 labWorks/lab1/calculation.py
PS D:\VS codes\dotnet lab>
```

12. To check the commits performed in the past


```
PS D:\VS codes\dotnet lab> git log
commit b9e81c114f2b3ec5cc2b0d52d0b08d436ed7b085 (HEAD -> main, origin/feature1, feature1)
Author: sumina <suminaawa123@gmail.com>
Date:   Sat Mar 22 18:13:23 2025 +0545

    Changes in New Branch

commit 342d257df4208ef35d379dd9a4b2dc1a7f6a753f
Author: sumina <suminaawa123@gmail.com>
Date:   Sat Mar 22 18:12:15 2025 +0545

    Changes in New Branch

commit 506de11e8fd5c1202e83922441db340d4f56d750
Author: sumina <suminaawa123@gmail.com>
Date:   Sat Mar 22 18:09:43 2025 +0545

    Changes in New Branch

commit 3c188b603c9c304de4c325ad3217bf5fd301ae85 (origin/main)
Author: sumina <suminaawa123@gmail.com>
commit b9e81c114f2b3ec5cc2b0d52d0b08d436ed7b085 (HEAD -> main, origin/feature1, feature1)
Author: sumina <suminaawa123@gmail.com>
```

13. Merging the branch in the GUI GitHub (Web)



Compare changes

Compare changes across branches, commits, tags, and more below. If you need to, you can also compare across forks.

base: main ← compare: main

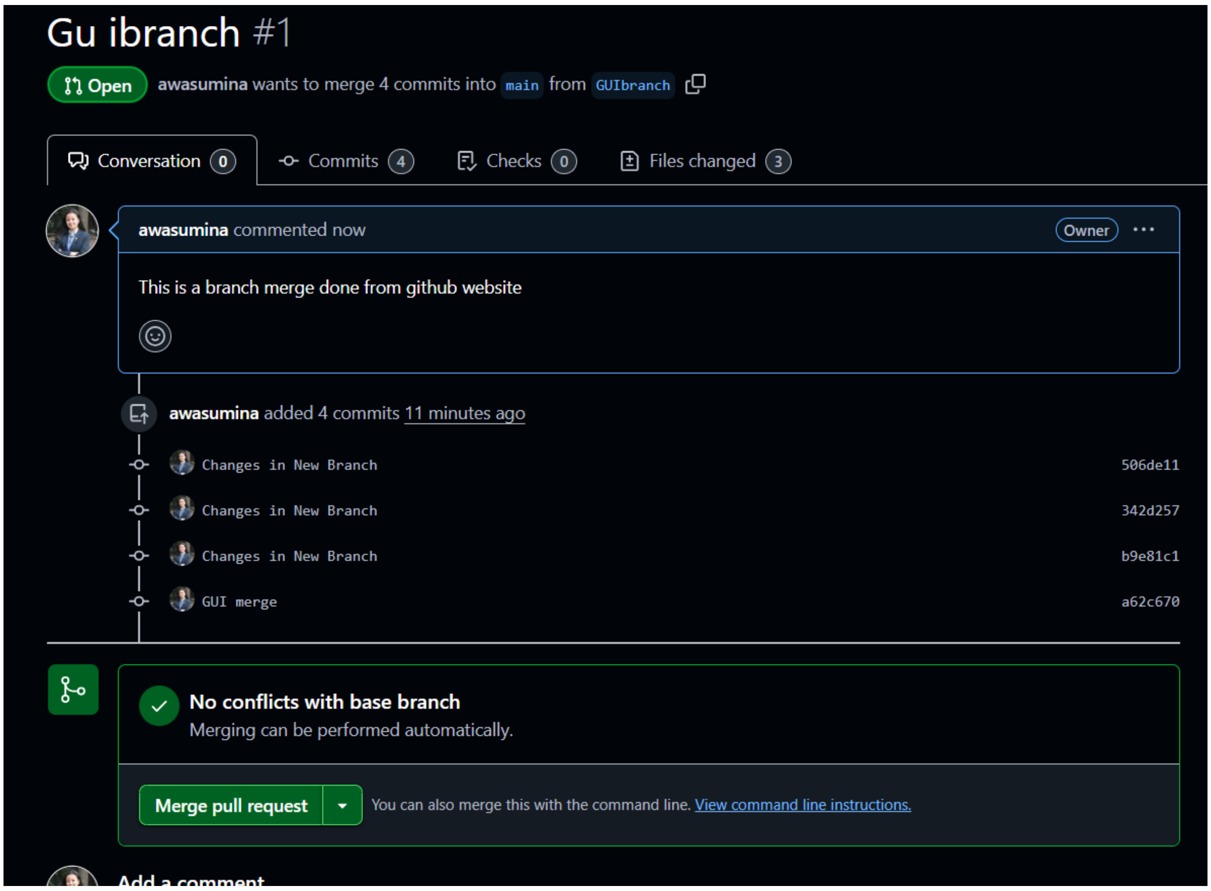
Choose different branches or forks above to discuss and review changes.
Learn about pull requests

Create pull request

Compare and review just about anything

Branches, tags, commit ranges, and time ranges. In the same repository and across forks.

Example comparisons	
GUIbranch	*now*
feature1	*6 minutes ago*
main@{1day}...main	*24 hours ago*



Gu ibranch #1

Open awasumina wants to merge 4 commits into `main` from `GUIbranch`

Conversation 0 Commits 4 Checks 0 Files changed 3

awasumina commented now Owner

This is a branch merge done from github website

awasumina added 4 commits 11 minutes ago

```
Changes in New Branch                                  506de11
Changes in New Branch                                  342d257
Changes in New Branch                                  b9e81c1
GUI merge                                              a62c670
```

No conflicts with base branch
Merging can be performed automatically.

Merge pull request

You can also merge this with the command line. View command line instructions.

Add a comment

Conclusion:

In this lab, we learn about the basics of the Git and GitHub. We perform initialization, branching, merging, pushing and commit.