



DEPARTMENT OF  
COMPUTER SCIENCE AND ENGINEERING

---

Title: Git

---

INTEGRATED DESIGN PROJECT II  
CSE 406



GREEN UNIVERSITY OF BANGLADESH

---

# 1 Objective(s)

- To learn about version control system
- Allow multiple developers to work on the same codebase simultaneously without interfering with each other's work.
- Keep track of all changes made to the codebase.

## 2 Introduction

### Why do we use git?

Git is a widely used version control system that offers many benefits for software development teams.

- **Version control:** Git allows developers to keep track of changes made to the codebase over time. This ensures that there is always a record of what changes were made, who made them, and when. This makes it easy to collaborate on code and troubleshoot problems that arise.
- **Collaboration:** Git allows multiple developers to work on the same codebase simultaneously without interfering with each other's work. This promotes collaboration and teamwork.
- **Branching:** Git allows developers to create multiple branches of the codebase. This allows developers to work on different features or bug fixes in parallel without affecting the main codebase.
- **Security:** Git allows developers to control access to the codebase and restrict certain users from making changes. This helps to prevent unauthorized changes or errors that could harm the codebase.
- **Backup and recovery:** Git provides a secure backup of the codebase in case of data loss or corruption. This makes it easier to recover from errors or mistakes that could harm the codebase.

## 3 Implementation

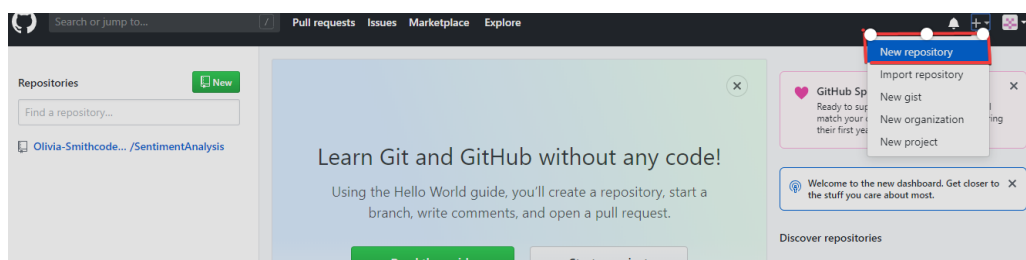
In this tutorial, you learn how to:

- Create a new repository
- Start and manage a new branch
- Change a file and Commit changes
- Open and merge a pull request

### 3.1 Using Command line to PUSH to GitHub

#### 3.1.1 Creating a new repository

1. You need to create a new repository and click on the plus sign.
2. Fill up all the required details, i.e., repository name, description and also make the repository public this time as it is free.



**Create a new repository**

A repository contains all project files, including the revision history. Already have a project repository? [Import a repository.](#)

Owner: Olivia-Smithcoder100

Repository name \*: FaceDetection ✓

Great repository names are short and memorable. Need inspiration? How about [animated-octo-men](#)

Description (optional)

Public: Anyone can see this repository. You choose who can commit.

Private: You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

☐ Initialize this repository with a README  
This will let you immediately clone the repository to your computer.

Add .gitignore: None | Add a license: None ⓘ

**Create repository**

### 3.1.2 Open your Git Bash

- Git Bash can be downloaded in [here](#) , and it is a shell used to interface with the operating system which follows the UNIX command.

### 3.1.3 Create your local project in your desktop directed towards a current working directory

- pwd stands for 'print working directory', which is used to print the current directory.
- Move to the specific path in your local computer by cd 'path\_name'. The cd commands stand for 'change directory' and it is used to change to the working directory in your operating system, and to locate your file, 'path\_name', i.e., C:/Users/Dell/Downloads/FaceDetect-master needs to be given. This command can identify the required file that you are looking to work with.

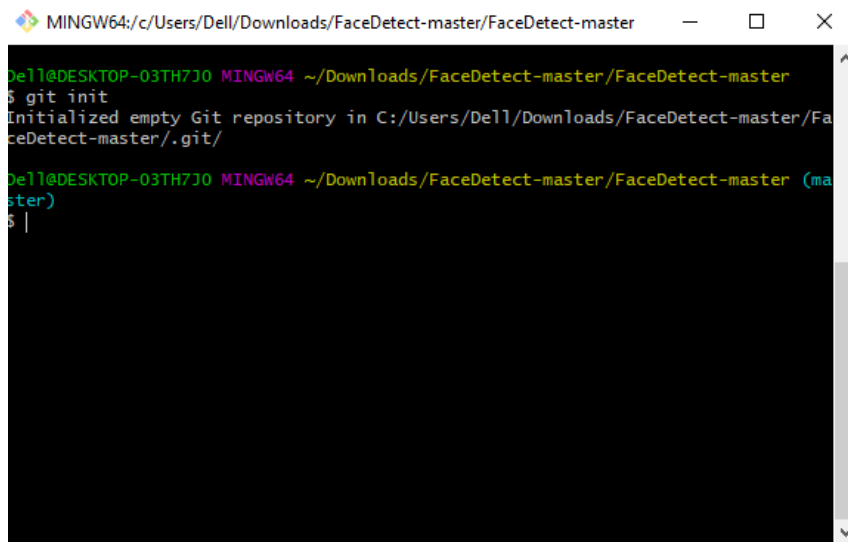
```
MINGW64:/c:/Users/Dell/Downloads/FaceDetect-master
Dell@DESKTOP-03TH730 MINGW64 ~
$ pwd
/c:/Users/Dell

Dell@DESKTOP-03TH730 MINGW64 ~
$ cd C:/Users/Dell/Downloads/FaceDetect-master

Dell@DESKTOP-03TH730 MINGW64 ~/Downloads/FaceDetect-master
$
```

### 3.1.4 Initialize the git repository

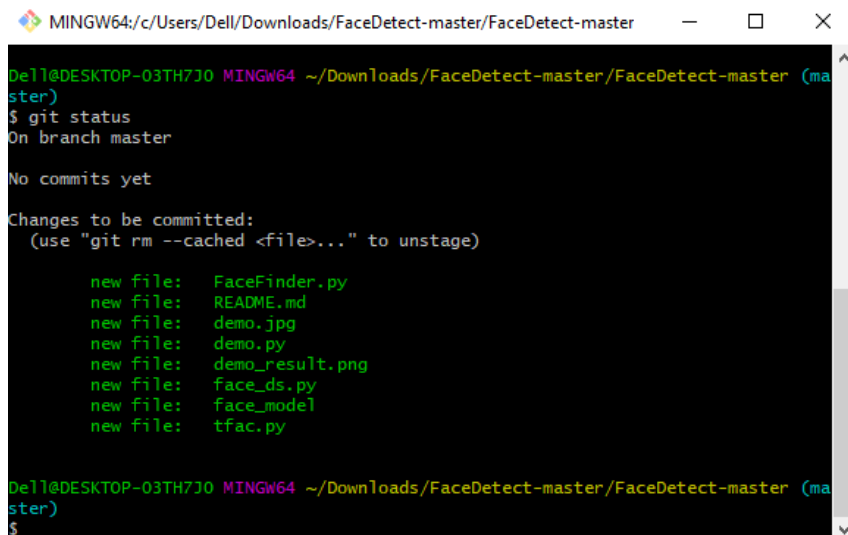
- Use **git init** to initialize the repository. It is used to create a new empty repository or directory consisting of files with the hidden directory. '.git' is created at the top level of your project, which places all of the revision information in one place.



```
MINGW64:/c:/Users/Dell/Downloads/FaceDetect-master/FaceDetect-master
Dell@DESKTOP-03TH7J0 MINGW64 ~/Downloads/FaceDetect-master/FaceDetect-master
$ git init
Initialized empty Git repository in C:/Users/Dell/Downloads/FaceDetect-master/FaceDetect-master/.git/
Dell@DESKTOP-03TH7J0 MINGW64 ~/Downloads/FaceDetect-master/FaceDetect-master (master)
$ |
```

### 3.1.5 Add the file to the new local repository

- Use **git add .** in your bash to add all the files to the given folder.
- Use **git status** in your bash to view all the files which are going to be staged to the first commit.



```
MINGW64:/c:/Users/Dell/Downloads/FaceDetect-master/FaceDetect-master
Dell@DESKTOP-03TH7J0 MINGW64 ~/Downloads/FaceDetect-master/FaceDetect-master (master)
$ git status
On branch master

No commits yet

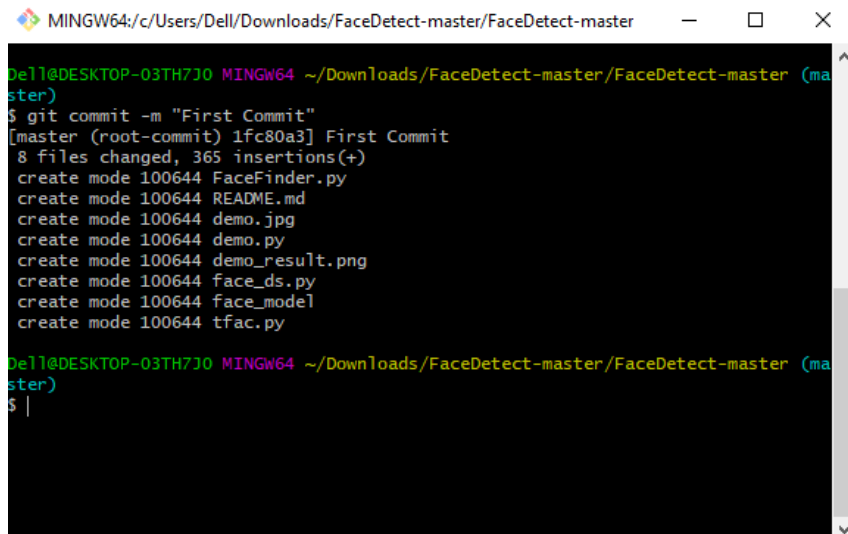
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   FaceFinder.py
        new file:   README.md
        new file:   demo.jpg
        new file:   demo.py
        new file:   demo_result.png
        new file:   face_ds.py
        new file:   face_model
        new file:   tfac.py

Dell@DESKTOP-03TH7J0 MINGW64 ~/Downloads/FaceDetect-master/FaceDetect-master (master)
$
```

### 3.1.6 Commit the files staged in your local repository by writing a commit message

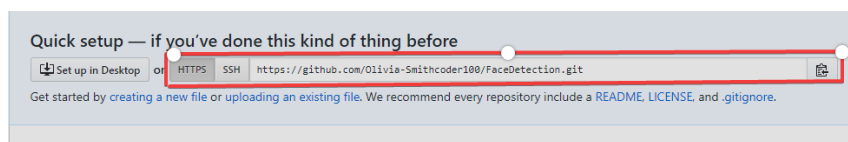
- You can create a commit message by **git commit -m 'your message'**, which adds the change to the local repository.
- **git commit** uses '-m' as a flag for a message to set the commits with the content where the full description is included, and a message is written in an imperative sentence up to 50 characters long and defining "what was changed", and "why was the change made".



```
MINGW64/c/Users/Dell/Downloads/FaceDetect-master/FaceDetect-master
Dell@DESKTOP-03TH7J0 MINGW64 ~/Downloads/FaceDetect-master/FaceDetect-master (master)
$ git commit -m "First Commit"
[master (root-commit) 1fc80a3] First Commit
8 files changed, 365 insertions(+)
create mode 100644 FaceFinder.py
create mode 100644 README.md
create mode 100644 demo.jpg
create mode 100644 demo.py
create mode 100644 demo_result.png
create mode 100644 face_ds.py
create mode 100644 face_model
create mode 100644 tfac.py
Dell@DESKTOP-03TH7J0 MINGW64 ~/Downloads/FaceDetect-master/FaceDetect-master (master)
$
```

### 3.1.7 Copy your remote repository's URL from GitHub

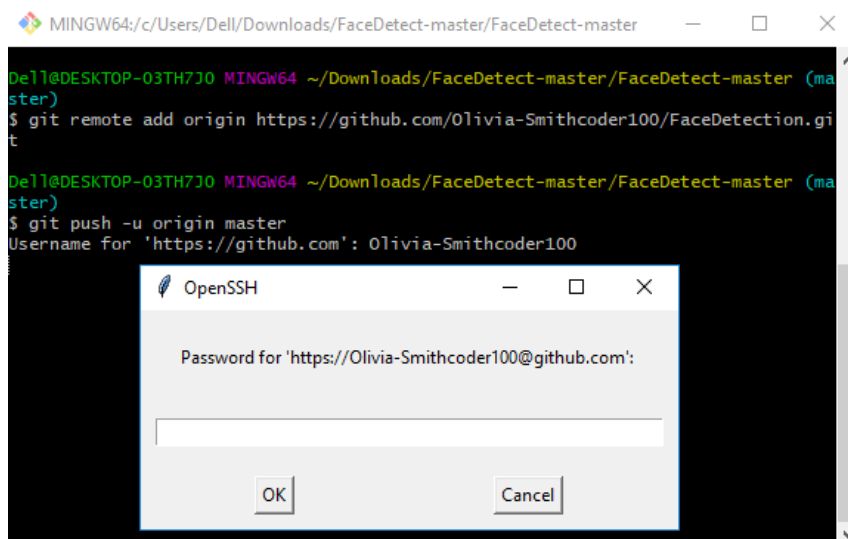
- The HTTPS or URL is copied from the given GitHub account, which is the place of the remote repository.



### 3.1.8 Add the URL copied, which is your remote repository to where your local content from your repository is pushed

- `git remote add origin 'your_url_name'`
- In the above code, The 'origin' is the remote name, and the remote URL is "https://github.com/Olivia-Smithcoder100/FaceDetection.git". You can see the remote as GitHub in this case, and GitHub provides the URL for adding to the remote repository.

### 3.1.9 Push the code in your local repository to GitHub



```
MINGW64/c/Users/Dell/Downloads/FaceDetect-master/FaceDetect-master
Dell@DESKTOP-03TH7J0 MINGW64 ~/Downloads/FaceDetect-master/FaceDetect-master (master)
$ git remote add origin https://github.com/Olivia-Smithcoder100/FaceDetection.git
Dell@DESKTOP-03TH7J0 MINGW64 ~/Downloads/FaceDetect-master/FaceDetect-master (master)
$ git push -u origin master
Username for 'https://github.com': Olivia-Smithcoder100
```

OpenSSH

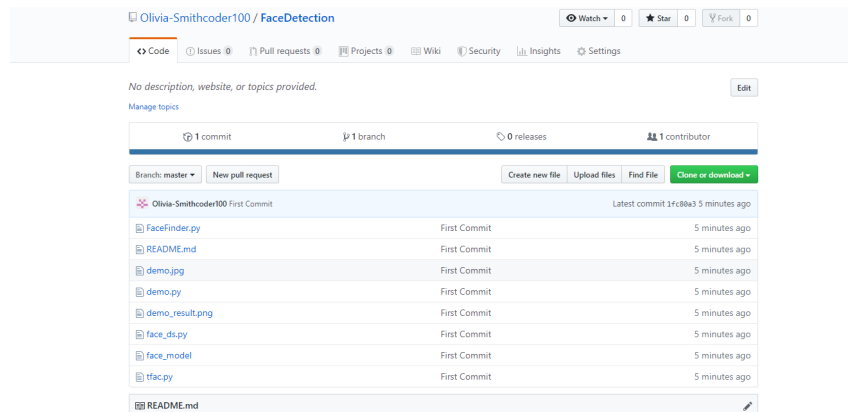
Password for 'https://Olivia-Smithcoder100@github.com':

OK Cancel

- `git push -u origin master` is used for pushing local content to GitHub.
- In the code, the origin is your default remote repository name and '-u' flag is upstream, which is equivalent to 'set-upstream.' and the master is the branch, name.upstream is the repository that we have cloned the project.
- Fill in your GitHub username and password.

### 3.1.10 View your files in your repository hosted on GitHub

- You can finally see the file hosted on GitHub.



## 3.2 PULL Request

If you make a change in a repository, GIT PULL can allow others to view the changes. It is used to acknowledge the change that you've made to the repository that you're working on. Or also called a target repository.

The simple command to PULL from a branch is:

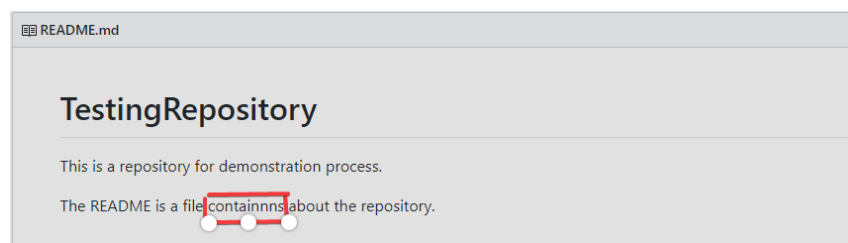
`git pull 'remote_name' 'branch_name'.`

The git pull command is a combination of git fetch which fetches the recent commits in the local repository and git merge, which will merge the branch from a remote to a local branch also 'remote\_name' is the repository name and 'branch\_name' is the name of the specific branch.

You'll be looking at two different ways on how to use the PULL request.

## 3.3 PULL Request through Command Line

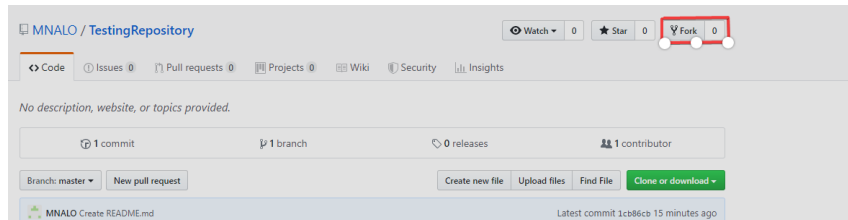
You can see the README files below which contains a typo. The README file has the word "contain" misspelled as "containnns". The owner of this repository is MNALO, and Olivia is the collaborator. She will solve the error and submit a PULL Request You'll see the process for making a PULL Request through a particular example given below.



In the file above, you can see a typo in the word "containnns".

### 3.3.1 Fork the Repository

- "The "Fork" is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.([Source](#))



### 3.3.2 Open your bash in your computer

- You need to move to the required path or folder by using the `cd` command, and the content can be viewed by using the `ls` command, which will list all of the present files in the directory and in our case you can see the 'README.md' is present.

```
MINGW64/c/Users/Dell/TestingRepository
Dell@DESKTOP-03TH7J0 MINGW64 ~
$ cd TestingRepository/
Dell@DESKTOP-03TH7J0 MINGW64 ~/TestingRepository (master)
$ ls
README.md
Dell@DESKTOP-03TH7J0 MINGW64 ~/TestingRepository (master)
$
```

### 3.3.3 Make a new branch

- You can create a new branch by using the `git checkout -b 'branch_name'`. In the above code, '-b' flag is used to create a new branch, and 'branch\_name' is used to give the branch a specific name, and with checkout, the branch is switched to the newly created branch.

```
Dell@DESKTOP-03TH7J0 MINGW64 ~/TestingRepository (master)
$ git checkout -b fix-typo-readme
Switched to a new branch 'fix-typo-readme'
```

### 3.3.4 Make a change by using vim from bash or direct replacement from the original README file

- You can change the word "containnns" to "contains" in the README file, and the changes with the current status can be viewed by using the following command.

```

Dell@DESKTOP-03TH7J0 MINGW64 ~/TestingRepository (fix-typo-readme)
$ git status
On branch fix-typo-readme
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   README.md

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

Dell@DESKTOP-03TH7J0 MINGW64 ~/TestingRepository (fix-typo-readme)
$ git diff
diff --git a/README.md b/README.md
index bc04944..48fb41f 100644
--- a/README.md
+++ b/README.md
@@ -1,4 +1,4 @@
 # TestingRepository
 This is a repository for demonstration process.
-The README is a file containnns about the repository.
+The README is a file contains about the repository.

```

### 3.3.5 Adding and Committing a file to the repository

You need to add and commit by the following commands.

```

Dell@DESKTOP-03TH7J0 MINGW64 ~/TestingRepository (fix-typo-readme)
$ git add README.md

Dell@DESKTOP-03TH7J0 MINGW64 ~/TestingRepository (fix-typo-readme)
$ git commit -m "Fix typo in README"
[fix-typo-readme 9f6cf3e] Fix typo in README
1 file changed, 1 insertion(+), 1 deletion(-)

Dell@DESKTOP-03TH7J0 MINGW64 ~/TestingRepository (fix-typo-readme)
$

```

### 3.3.6 Push the repository to the GitHub

- You need to push the content by git push origin 'branch\_name'
- In the above code, the origin is the remote repository, and 'branch\_name' is the required branch that you need to upload your local content.

```

Dell@DESKTOP-03TH7J0 MINGW64 ~/TestingRepository (fix-typo-readme)
$ git push origin fix-typo-readme
Username for 'https://github.com': Olivia-Smithcoder100
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 304 bytes | 152.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'fix-typo-readme' on GitHub by visiting:
remote:   https://github.com/Olivia-Smithcoder100/TestingRepository/pull/new/fix-typo-readme
remote:
To https://github.com/Olivia-Smithcoder100/TestingRepository.git
 * [new branch]      fix-typo-readme -> fix-typo-readme

Dell@DESKTOP-03TH7J0 MINGW64 ~/TestingRepository (fix-typo-readme)
$ |

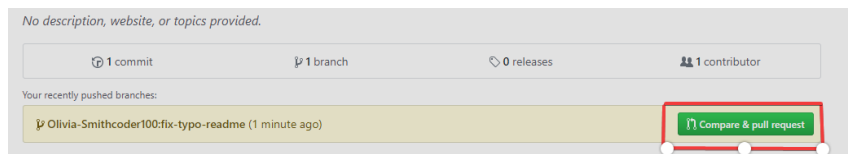
```

### 3.3.7 PULL request for a specific branch on GitHub

- You can move to your repository in GitHub and see that there is a new branch.
- You can now move to step 8, but there is a need for a local repository update with the upstream repository, read this detailed blog on [How To Create a Pull Request on GitHub](#)

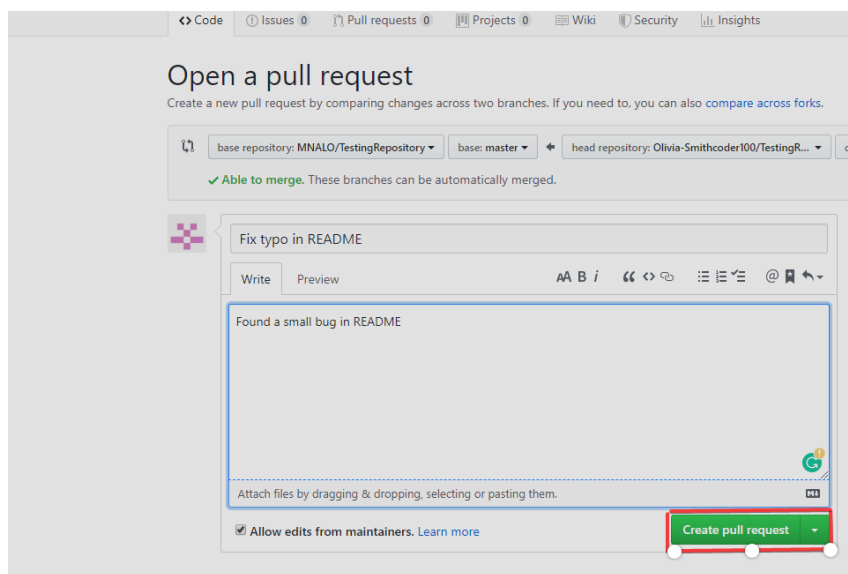


- Alternatively, you can do git pull-request in the command line and complete the PULL Request to GitHub, where it will force push your current branch to a remote repository.



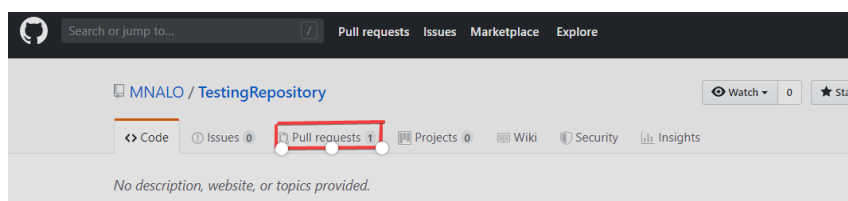
### 3.3.8 Open a Pull request

You need to click the button on "Create pull request," to finish the action.

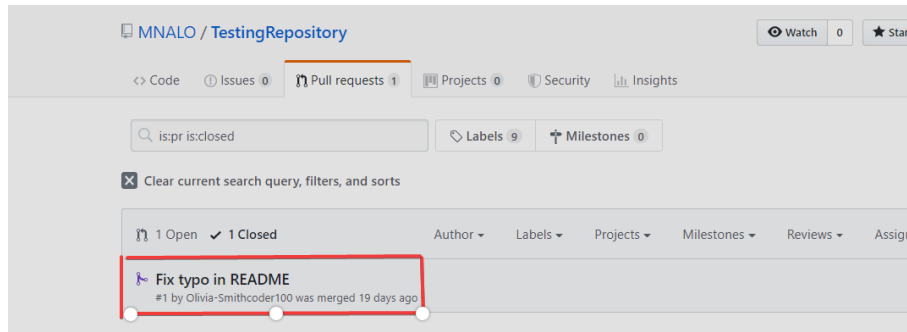


### 3.3.9 Deleting a Branch after the PULL Request is Merged

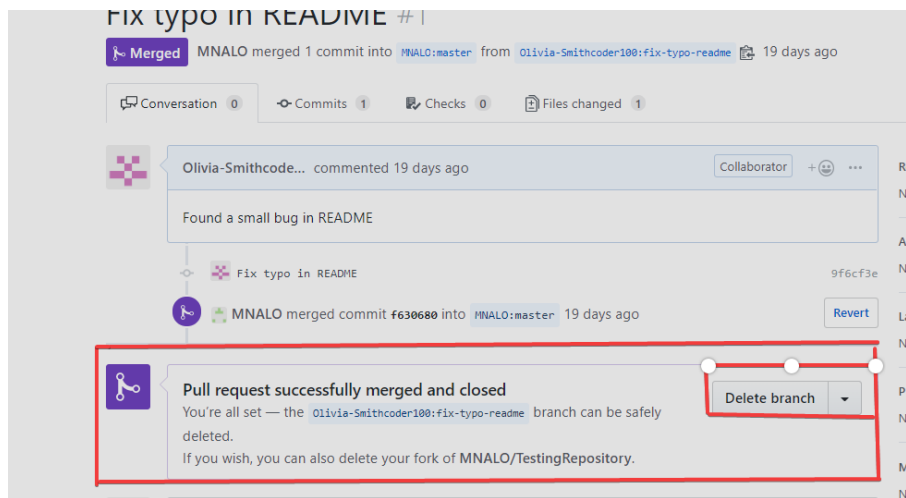
You need to move to the main page of the repository and click "Pull requests".



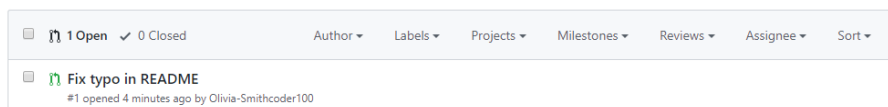
You need to click 'Closed' to see the lists of all the PULL Requests that you've made, but there is only one at the moment which needs to be selected. It is the one related to your branch that you want to delete.



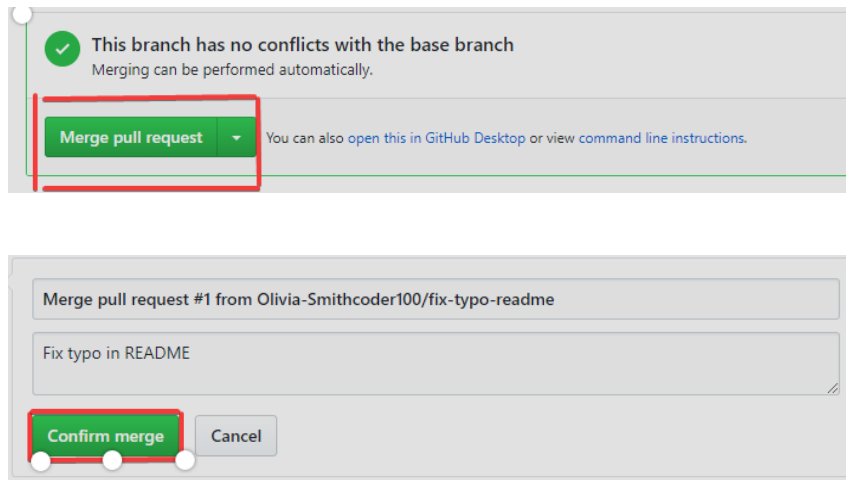
You can now click 'Delete branch' to complete the action.



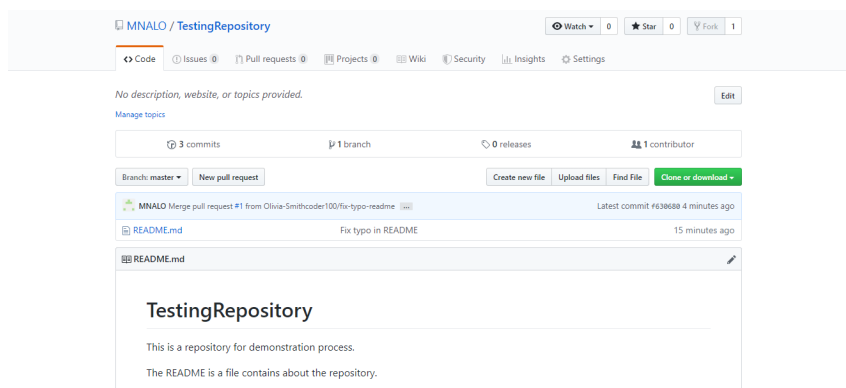
The owner of the repository can view all the commits, pull request, etc., made by collaborators and others. The changes made by someone can be significant, quick fixes for a bug, errors, etc., and are added to the project.



The owner now clicks "Merge pull request". Also, he/she will click "Confirm merge" through the following process.



The last change made to the README.md file with a corrected typo is below.



## 4 Discussion & Conclusion

After reading this tutorial on the version control system, Git, you would have understood how the VCS works. You will also be able to create a new repository and integrate it into any project for better management and collaboration.

## 5 Lab Task (Please implement yourself and show the output to the instructor)

1. Create a GitHub repository for the given project.
2. Implement the branching, committing, pull request and merge options to track and manage the changes of the given project using Git Bash.

## 6 Policy

Copying from internet, classmate, seniors, or from any other source is strongly prohibited. 100% marks will be *deducted* if any such copying is detected.