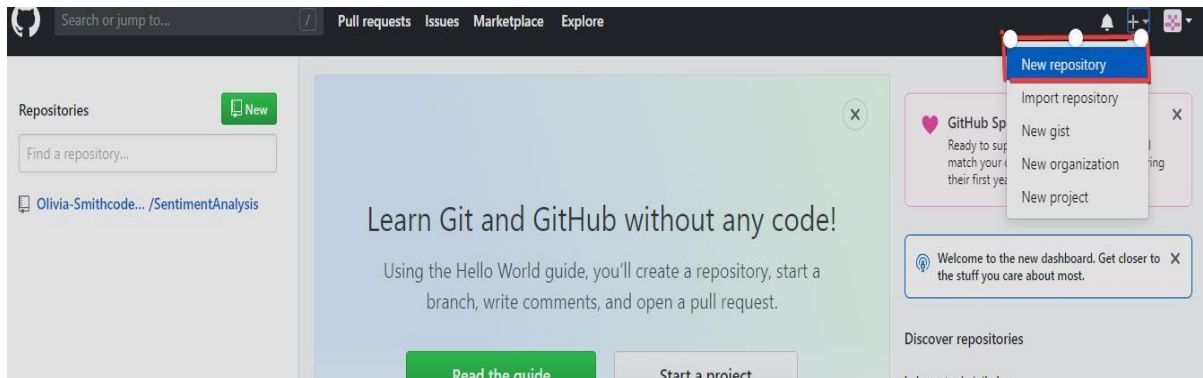


# Write step to push file to GitHub

## Using Command line to PUSH to GitHub

### 1. Creating a new repository

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

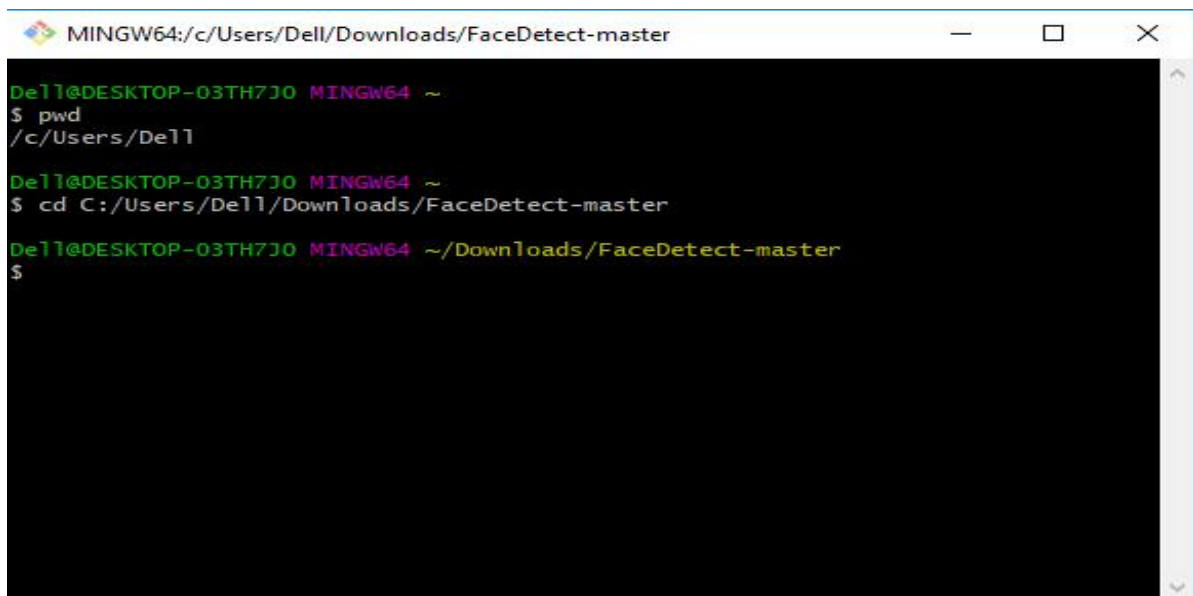
A screenshot of the 'Create a new repository' form on GitHub. The form is titled 'Create a new repository' and has a subtitle 'A repository contains all project files, including the revision history. Already have a project repository? [Import a repository.](#)'. The form has several sections: 'Owner' with a dropdown menu showing 'Olivia-Smithcoder100'; 'Repository name \*' with a text input field containing 'FaceDetection' (highlighted with a red box and a green checkmark); 'Description (optional)' with a text area; 'Visibility' with two radio buttons: 'Public' (selected, highlighted with a red box) and 'Private'; 'Skip this step if you're importing an existing repository.' with a checkbox for 'Initialize this repository with a README'; and 'Add .gitignore: None' and 'Add a license: None' dropdowns. At the bottom, there is a green 'Create repository' button (highlighted with a red box).

### 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. 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

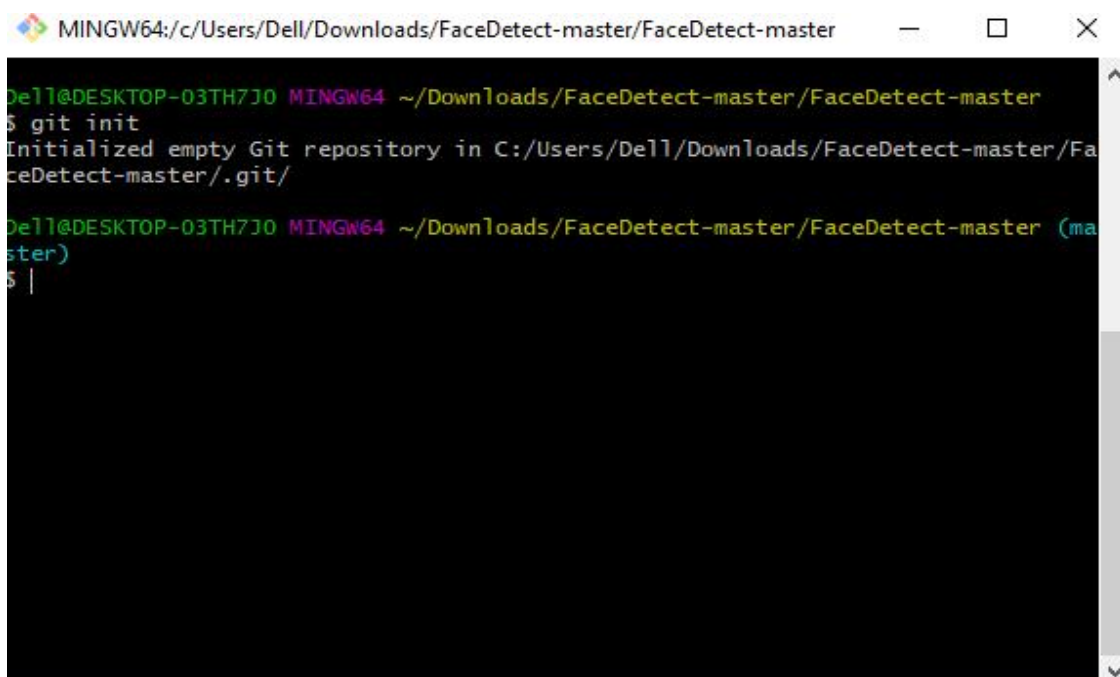
Dell@DESKTOP-03TH7J0 MINGW64 ~
$ pwd
/c/Users/Dell

Dell@DESKTOP-03TH7J0 MINGW64 ~
$ cd C:/Users/Dell/Downloads/FaceDetect-master

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

### 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.



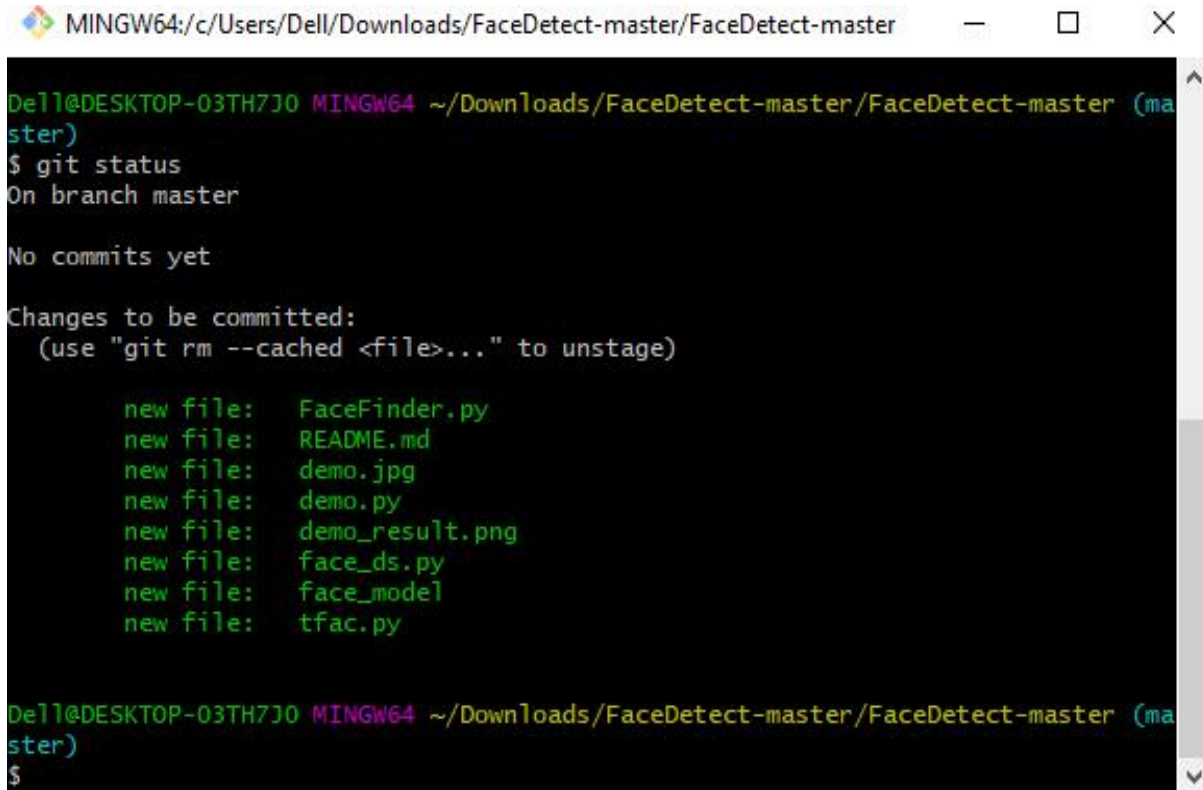
```
MINGW64:/c/Users/Dell/Downloads/FaceDetect-master/FaceDetect-master

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)
$ |
```

## 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

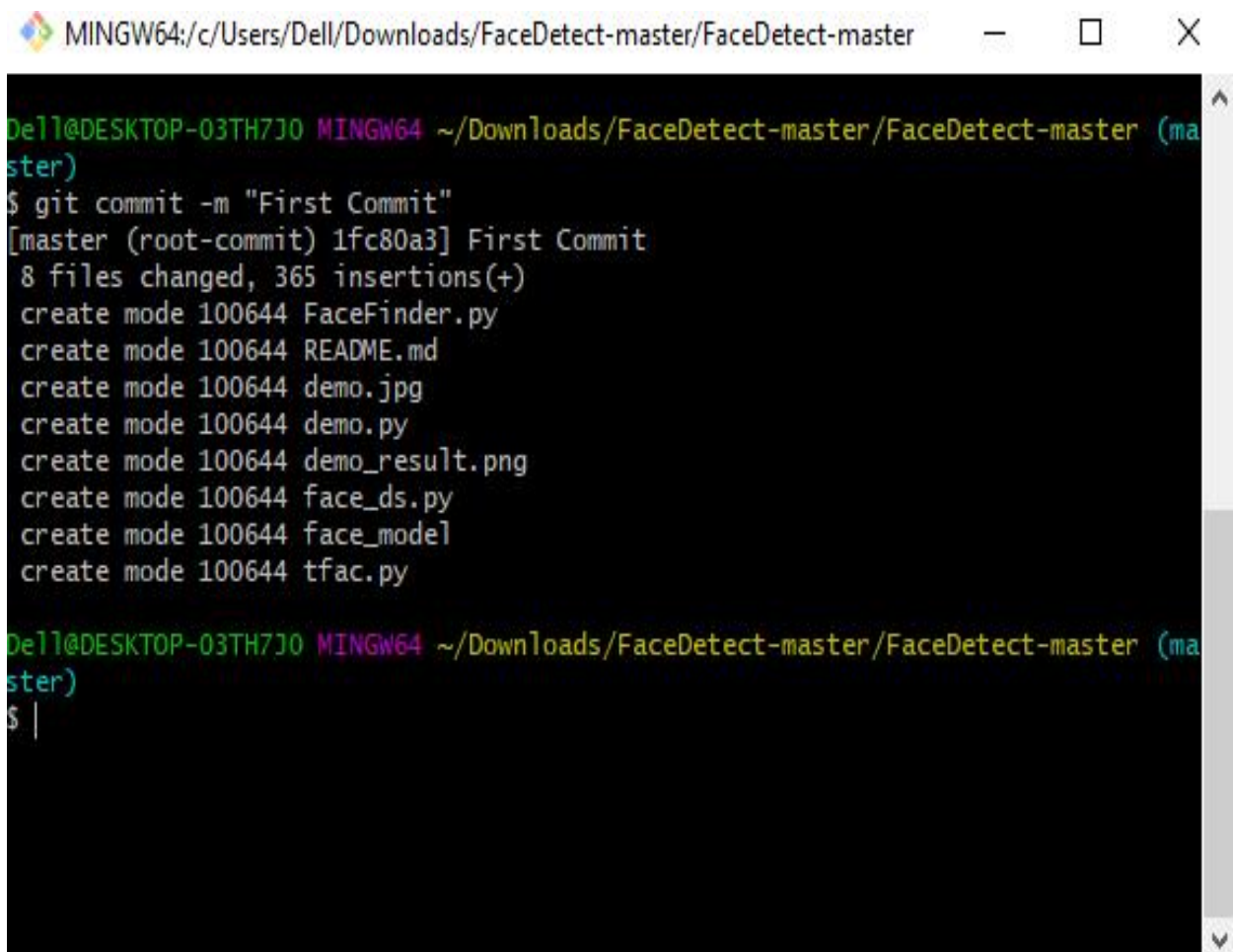
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

    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)
$
```

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

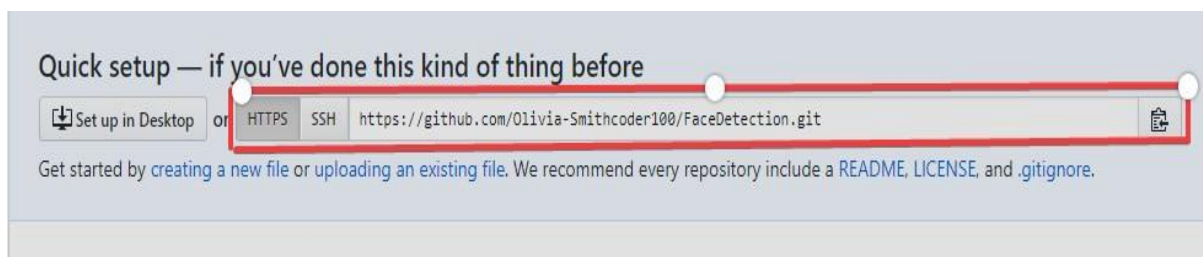
- You can create a commit message by `git commit -m 'your message'`, which adds the change to the local repository.
- `git commit` uses `-m` as a flag for a message to set the commits with the content where the full description is included, and a message is written in an imperative sentence up to 50 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)
$ |
```

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.

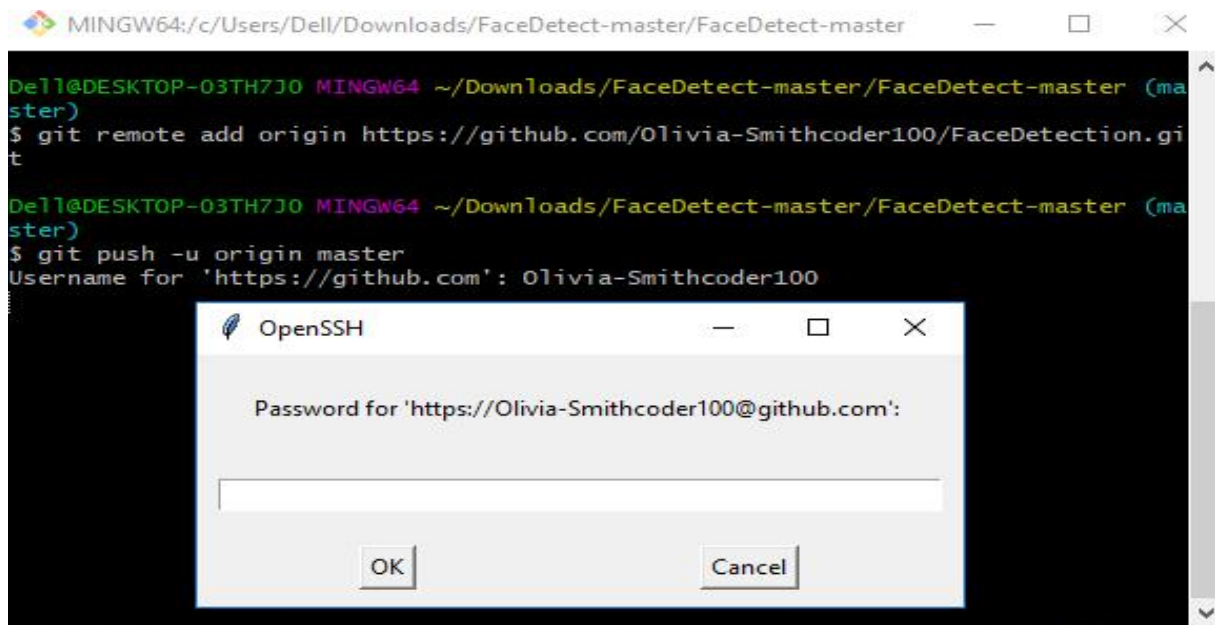


8. Add the URL copied, which is your remote repository to where your local content from your repository is pushed

- `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.

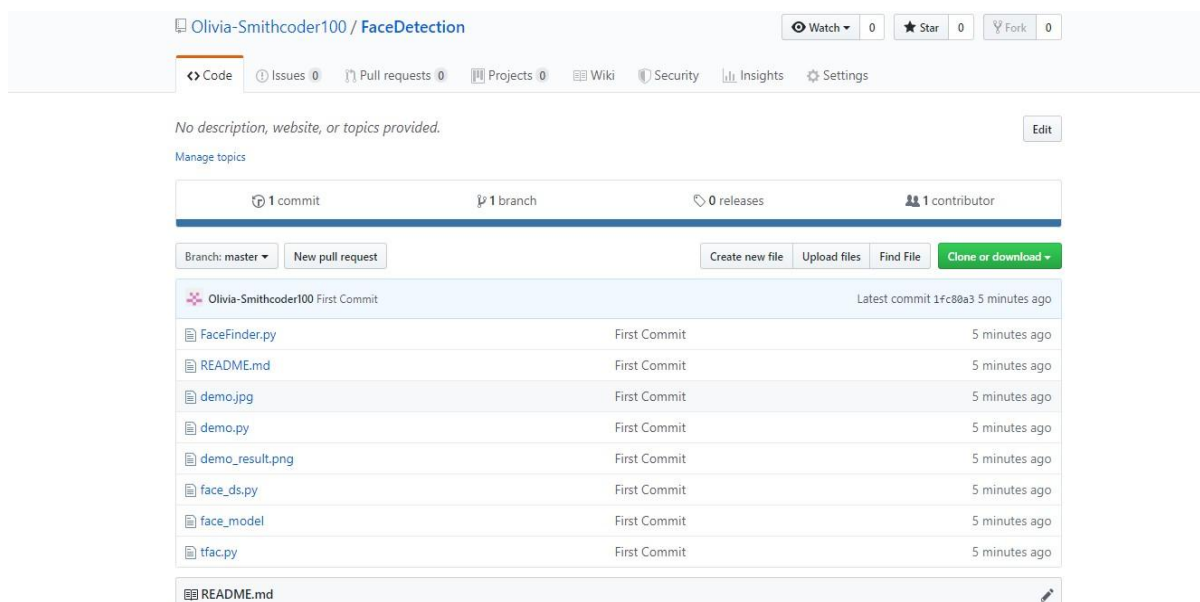
## 9. Push the code in your local repository to GitHub

- `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.



## 10. View your files in your repository hosted on GitHub

- You can finally see the file hosted on GitHub.



# Write step to pull a file from GitHub

## 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:

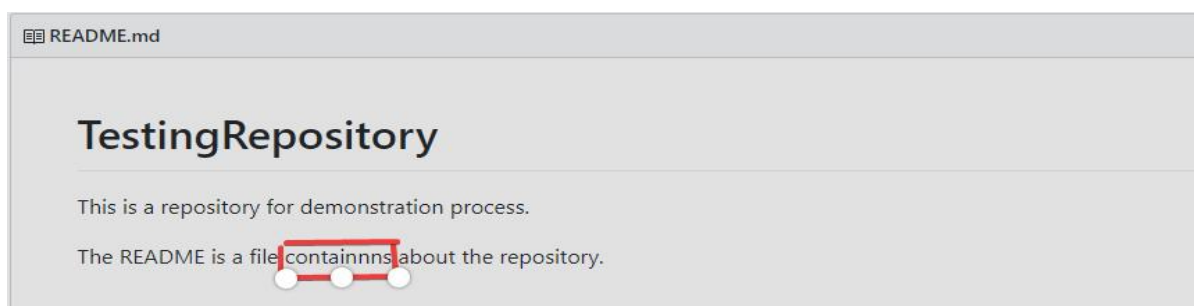
```
git pull 'remote_name' 'branch_name'.
```

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.

## PULL Request through Command Line

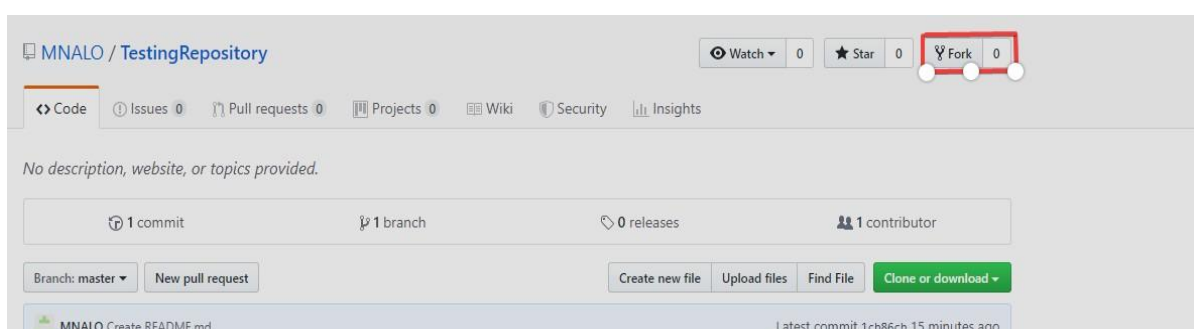
You can see the README files below which contains a typo. The README file has the word "contain" misspelled as "containnns". The owner of this repository is MNALO, and Olivia is the collaborator. She will solve the error and submit a PULL Request You'll see the process for making a PULL Request through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



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

### 1. Fork the Repository

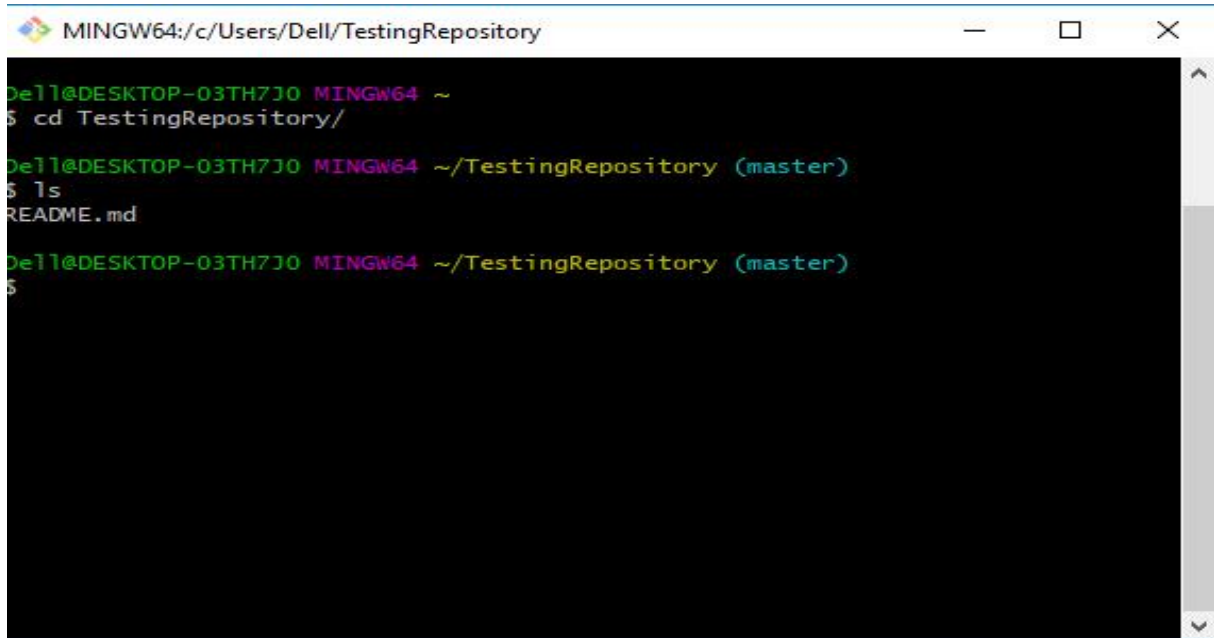
- "The "Fork" is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project." ([Source](#))





## 2. Open your bash in your computer

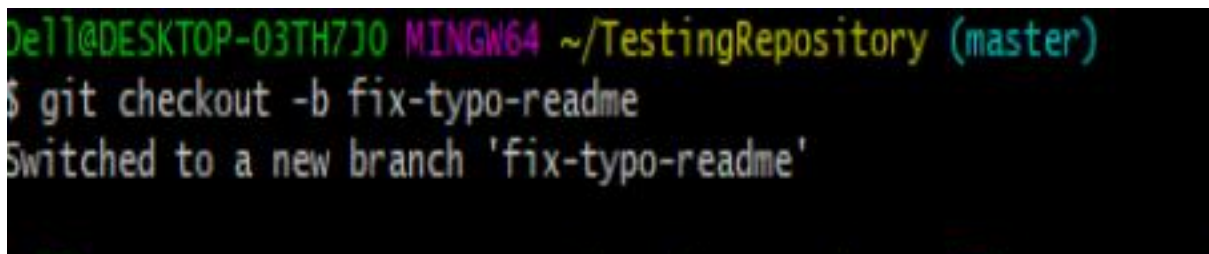
- You need to move to the required path or folder by using the `cd` command, and the content can be viewed by using the `ls` command, which will list all of the present files in the directory and in our case you can see the 'README.md' is present.



```
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. 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'
```

## 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.

```

## 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)
$

```



# Write a step to branch on the GitHub

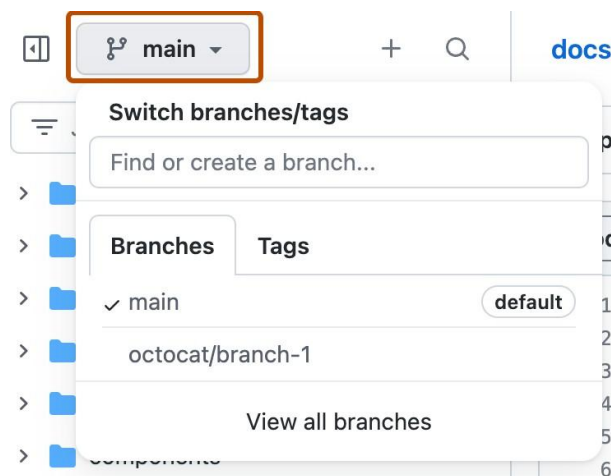
## [Creating a branch](#)

You can create a branch in different ways on GitHub.

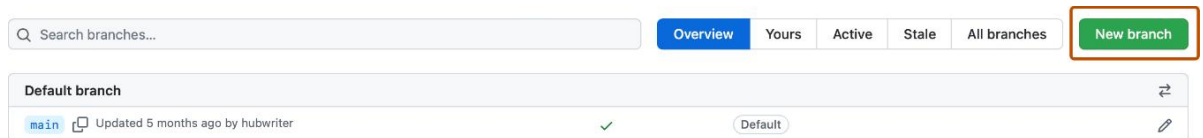
**Note:** You can only create a branch in a repository to which you have push access.

## [Creating a branch via the branches overview](#)

1. On GitHub.com, navigate to the main page of the repository.
2. From the file tree view on the left, select the  
  
2. branch dropdown menu, then click **View all branches**. You can also find the branch dropdown menu at the top of the integrated file editor.



3. Click **New branch**.

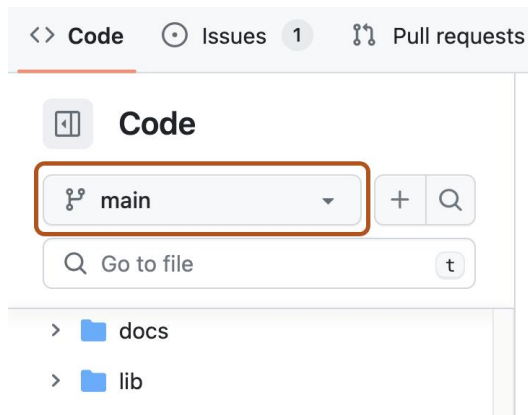


4. Under "Branch name", type a name for the branch.
5. Under "Branch source", choose a source for your branch.
  - If your repository is a fork, select the repository dropdown menu and click your fork or the upstream repository.
  - Select the branch dropdown menu and click a branch.
6. Click **Create branch**.

## [Creating a branch using the branch dropdown](#)

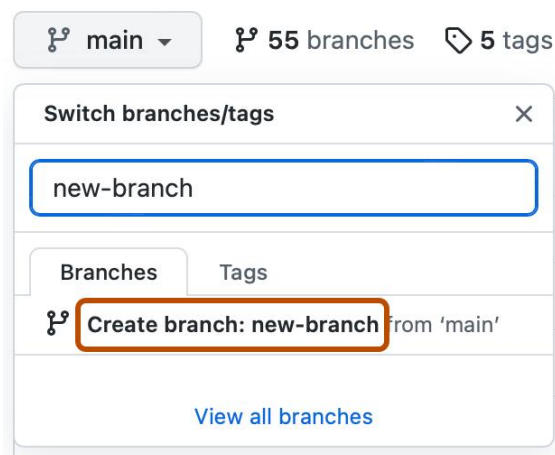
1. On GitHub.com, navigate to the main page of the repository.
2. Select the

□ branch dropdown menu, in the file tree view or at the top of the integrated file editor.



□ Optionally, if you want to create the new branch from a branch other than the default branch of the repository, click another branch, then select the branch dropdown menu again.

□ In the "Find or create a branch..." text field, type a unique name for your new branch, then click **Create branch**.

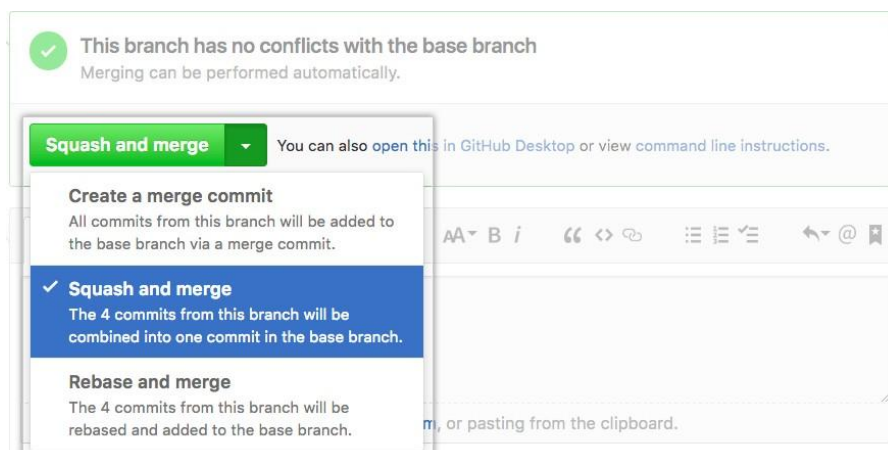


# Write a step to merge branches in GitHub

## Merge branches in GitHub

Now, let's discuss how to merge GitHub branches. It's relatively simple to do:

1. Navigate to your repository, then find and click the **Pull requests** button. It should be in between the Issues and Actions buttons. You'll see a summary of all pull requests you have pending.
2. Navigate to and click on the pull request you'd like to merge into the main branch.
3. Now, you have a few choices for initiating the merge, depending on your repository's merge options:
  - **Merge every commit into your main branch:**  
To do this, click **Merge pull request**. If this button does not appear, open the dropdown menu and click on **Create a merge commit**.
  - **Combine the commits into one large commit:**  
You can "squash" your commits together by clicking the dropdown menu, then choosing **Squash and merge** and clicking the new **Squash and merge** button. Doing this helps you create a cleaner, more streamlined Git history for your repository.
  - **Rebase each commit onto the main branch:**  
To do this, click the dropdown menu, choose **Rebase and merge**, and click the new **Rebase and merge** button. This also creates a cleaner project history.
4. You can now leave a comment if you'd like, or you can accept the default message GitHub provides you.
5. Scroll below the commit message field to choose a Git author email address.
6. Click on **Confirm merge**, **Confirm squash and merge**, or **Confirm rebase and merge**, depending on which option you choose in Step 3.
7. If the merge occurs successfully, GitHub will display a note stating so. This helps confirm that you're good to go.
8. If you'd like, you can now delete the branch by clicking **Delete branch** to keep things nice and neat. Alternatively, you can set it up so that the branch deletes automatically once you complete a merge to save some time.



The merge button

