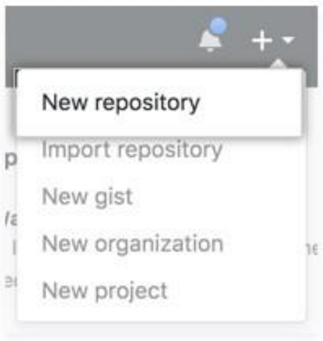
6. Git Hub

Adding a local repository to GitHub using Git

1. Create a new repository on GitHub.com. To avoid errors, do not initialize the new repository with *README*, license, or `gitignore` files. You can add these files after your project has been pushed to GitHub.



2. Open Git Bash.

3. Change the current working directory to your local project.

4. Use the `init` command to initialize the local directory as a Git repository. By default, the initial branch is called `master`.

   If you're using Git 2.28.0 or a later version, you can set the name of the default branch using `-b`.

   ```
   $ git init -b main
   ```

   If you're using Git 2.27.1 or an earlier version, you can set the name of the default branch using `&& git symbolic-ref HEAD refs/heads/main`.

   ```
   $ git init && git symbolic-ref HEAD refs/heads/main
   ```

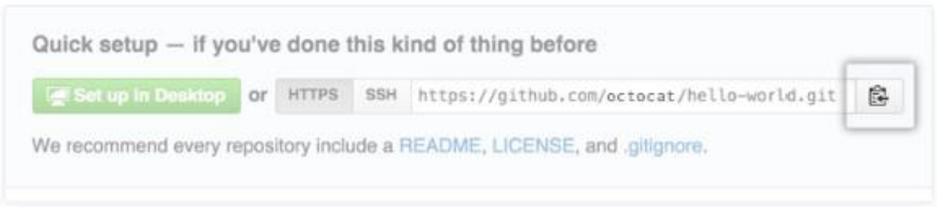
5. Add the files in your new local repository. This stages them for the first commit.

   ```
   $ git add .
   # Adds the files in the local repository and stages them for commit. To unstag
   ```

6. Commit the files that you've staged in your local repository.

```
$ git commit -m "First commit"
# Commits the tracked changes and prepares them to be pushed to a remote repos
```

7. At the top of your repository on GitHub.com's Quick Setup page, click 📋 to copy the remote repository URL.



Quick setup — if you've done this kind of thing before

https://github.com/octocat/hello-world.git

We recommend every repository include a README, LICENSE, and .gitignore.

8. In the Command prompt, add the URL for the remote repository where your local repository will be pushed.

```
$ git remote add origin <REMOTE_URL>
# Sets the new remote
$ git remote -v
# Verifies the new remote URL
```

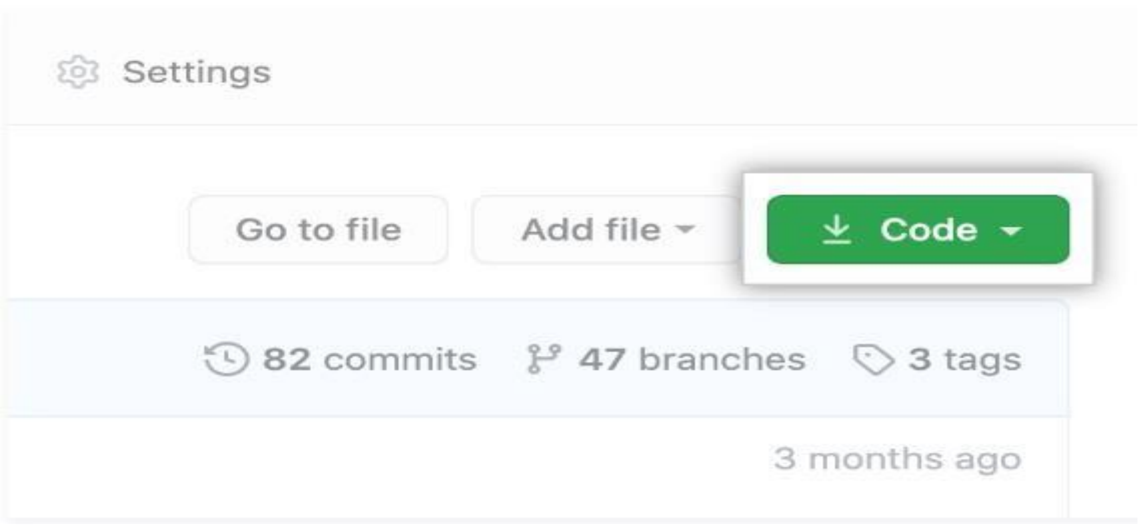
9. Push the changes in your local repository to GitHub.com.

```
$ git push origin main
# Pushes the changes in your local repository up to the remote repository you
```

Cloning a repository

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

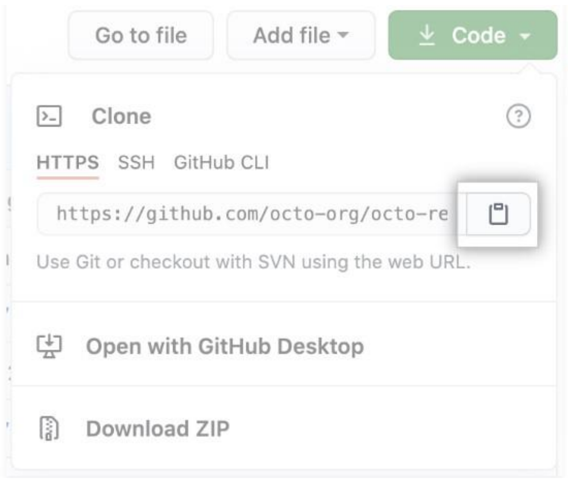
Above the list of files, click **Code**.



3. Copy the URL for the repository.

 To clone the repository using HTTPS, under "HTTPS", click . To clone the repository using an SSH key, including a certificate issued by your organization's SSH certificate authority, click **SSH**, then click .

To clone a repository using GitHub CLI, click **GitHub CLI**, then click .



https://github.com/octo-org/octo-re

Use Git or checkout with SVN using the web URL.

4. Open Git Bash.
5.Change the current working directory to the location where you want the cloned directory.
6.Type `git clone`, and then paste the URL you copied earlier.
Press **Enter** to create your local clone.

```
$ git clone https://github.com/YOUR-USERNAME/YOUR-REPOSITORY
```

7.Press **Enter** to create your local clone.

```
$ git clone https://github.com/YOUR-USERNAME/YOUR-REPOSITORY
> Cloning into `Spoon-Knife`...
> remote: Counting objects: 10, done.
> remote: Compressing objects: 100% (8/8), done.
> remove: Total 10 (delta 1), reused 10 (delta 1)
> Unpacking objects: 100% (10/10), done.
```

Making and Recording Changes

In order to begin tracking a new file, you use the command `git add`. To begin tracking the `README` file, you can run this:

```
$ git add README
```

If you run your status command again, you can see that your `README` file is now tracked and staged to be committed

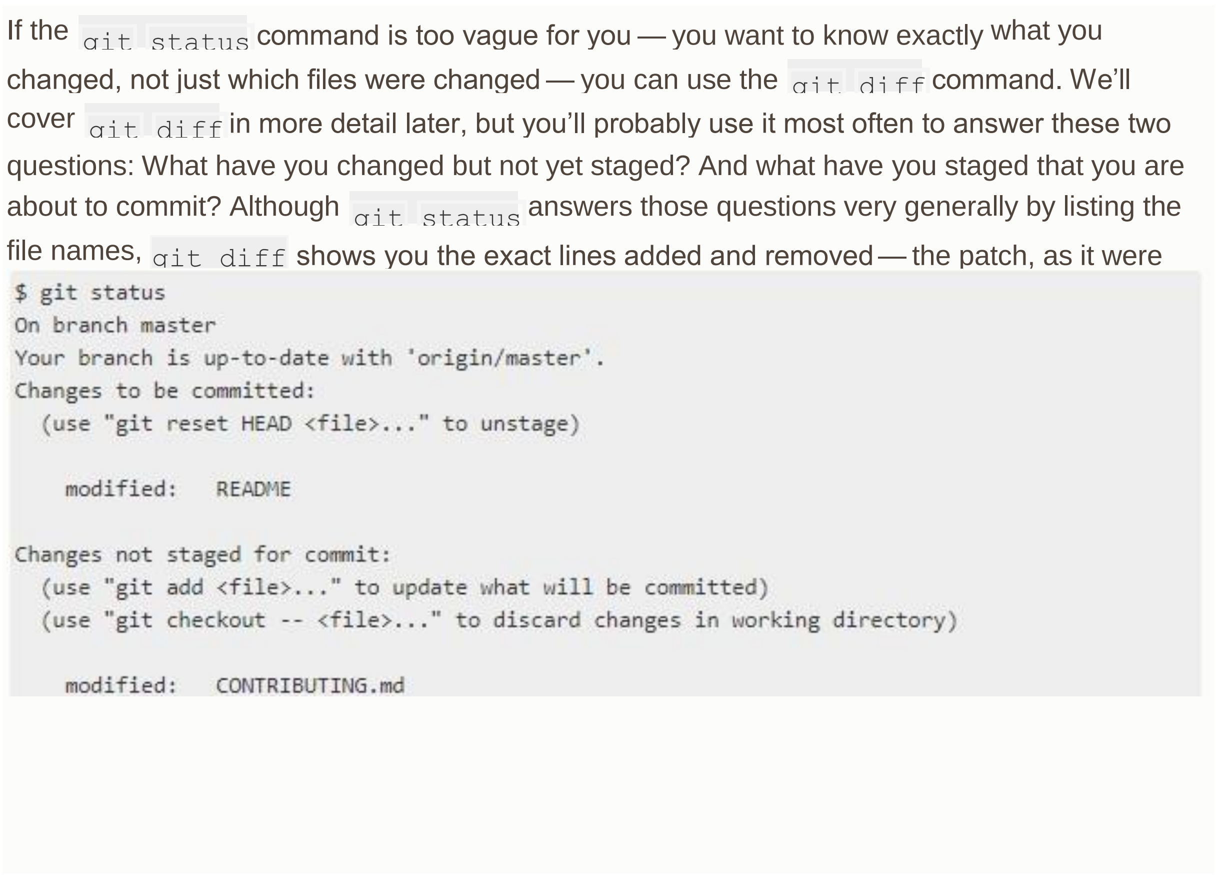
```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)

    new file:   README
```

Staging and Committing changes

Let's change a file that was already tracked. If you change a previously tracked file called `CONTRIBUTING.md` and then run your `git status` command again, you get something that looks like this:

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   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:   CONTRIBUTING.md
```

Viewing the Staged and Unstaged Changes



If the `git status` command is too vague for you — you want to know exactly what you changed, not just which files were changed — you can use the `git diff` command. We'll cover `git diff` in more detail later, but you'll probably use it most often to answer these two questions: What have you changed but not yet staged? And what have you staged that you are about to commit? Although `git status` answers those questions very generally by listing the file names, `git diff` shows you the exact lines added and removed — the patch, as it were

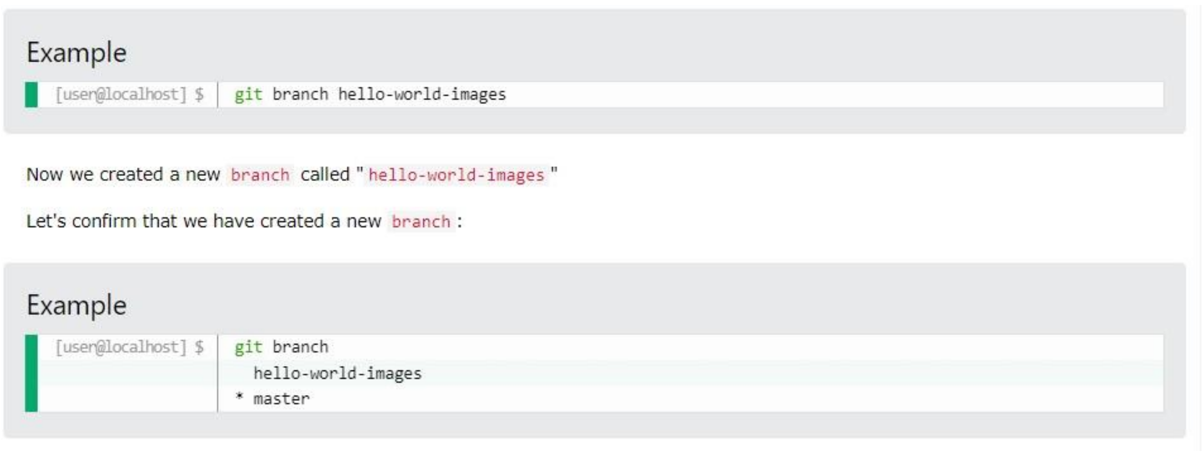
```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   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:   CONTRIBUTING.md
```

New Git Branch

Let add some new features to our `index.html` page.

We are working in our local repository, and we do not want to disturb or possibly wreck the main project.

So we create a new `branch`:



Example

```
[user@localhost] $   git branch hello-world-images
```

Now we created a new `branch` called "`hello-world-images`"

Let's confirm that we have created a new `branch`:

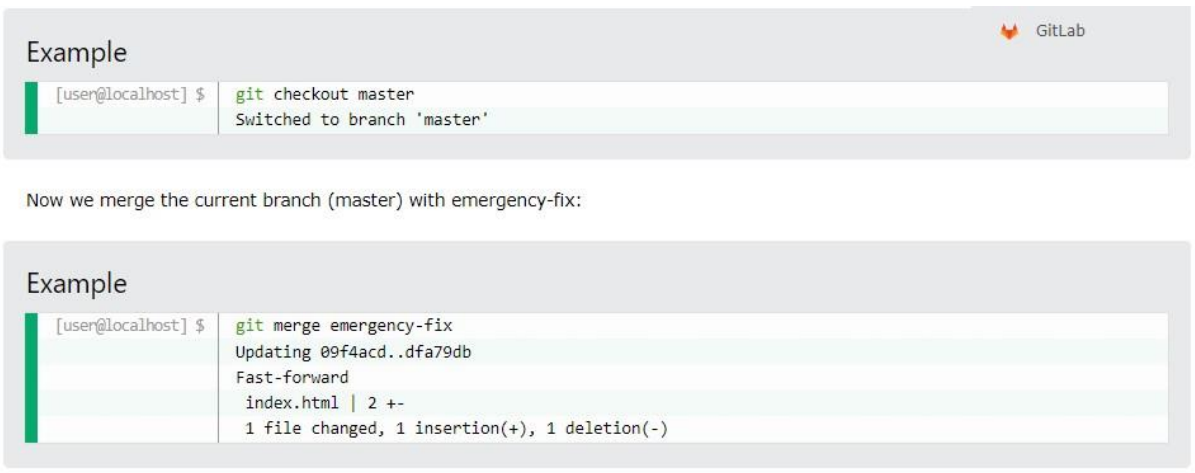
Example

```
[user@localhost] $   git branch
                       hello-world-images
                     * master
```

Merge Branches

We have the emergency fix ready, and so let's merge the master and emergency-fix branches.

First, we need to change to the master branch:



Example

```
[user@localhost] $   git checkout master
                     Switched to branch 'master'
```

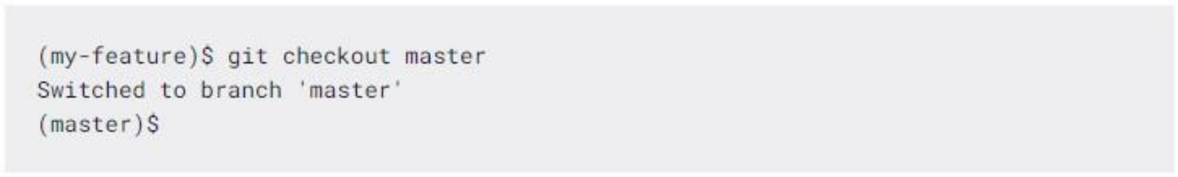
Now we merge the current branch (master) with emergency-fix:

Example

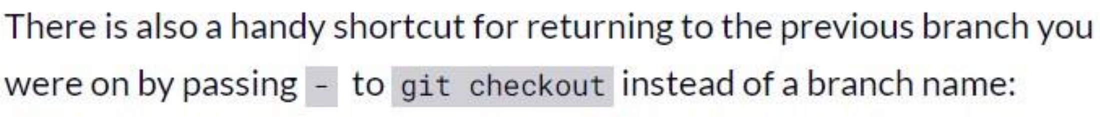
```
[user@localhost] $   git merge emergency-fix
                     Updating 09f4acd..dfa79db
                     Fast-forward
                      index.html | 2 +-
                      1 file changed, 1 insertion(+), 1 deletion(-)
```

How to switch to an existing branch in Git

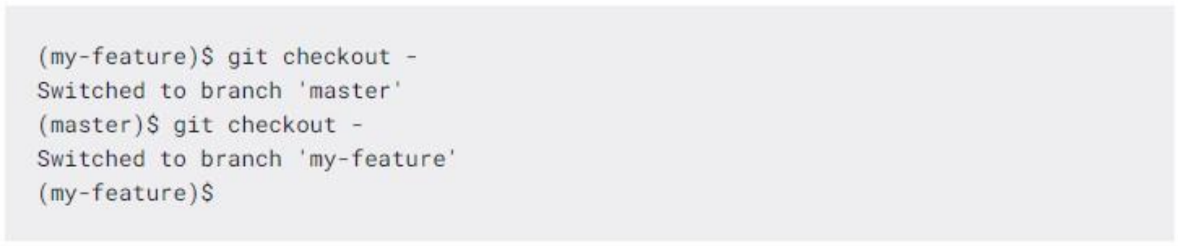
To switch to an existing branch, you can use `git checkout` again (without the `-b` flag) and pass the name of the branch you want to switch to:



```
(my-feature)$ git checkout master
Switched to branch 'master'
(master)$
```



There is also a handy shortcut for returning to the previous branch you were on by passing `-` to `git checkout` instead of a branch name:



```
(my-feature)$ git checkout -
Switched to branch 'master'
(master)$ git checkout -
Switched to branch 'my-feature'
(my-feature)$
```

Switching between Branches

Switch to a specified branch. The working tree and the index are updated to match the branch. All new commits will be added to the tip of this branch.

```
git switch [<options>] [--no-guess] <branch>
git switch [<options>] --detach [<start-point>]
git switch [<options>] (-c|-C) <new-branch> [<start-point>]
git switch [<options>] --orphan <new-branch>
```

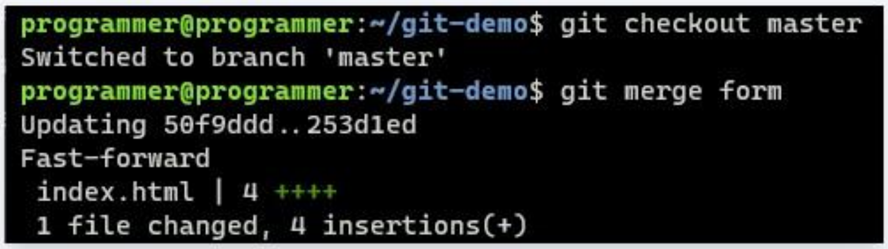
Merging Git Branches Locally

1. Run the `git checkout` command below to switch to the `master` branch since you'll update it with the code from the form branch.

```
git checkout master
```

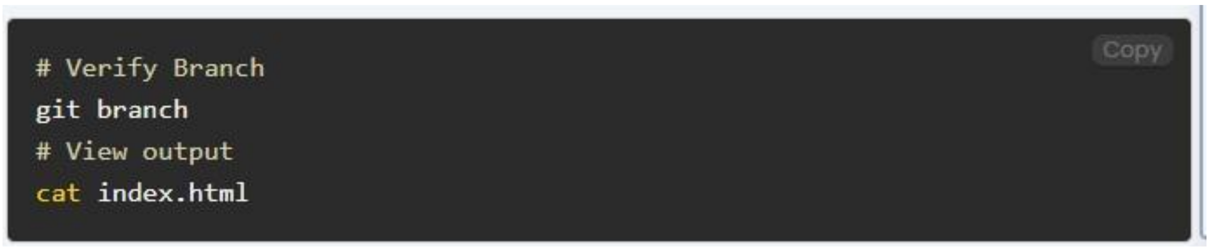
2. Run the commands below to combine your target branch (`form`) with the current branch (`master`).

```
git merge form
```

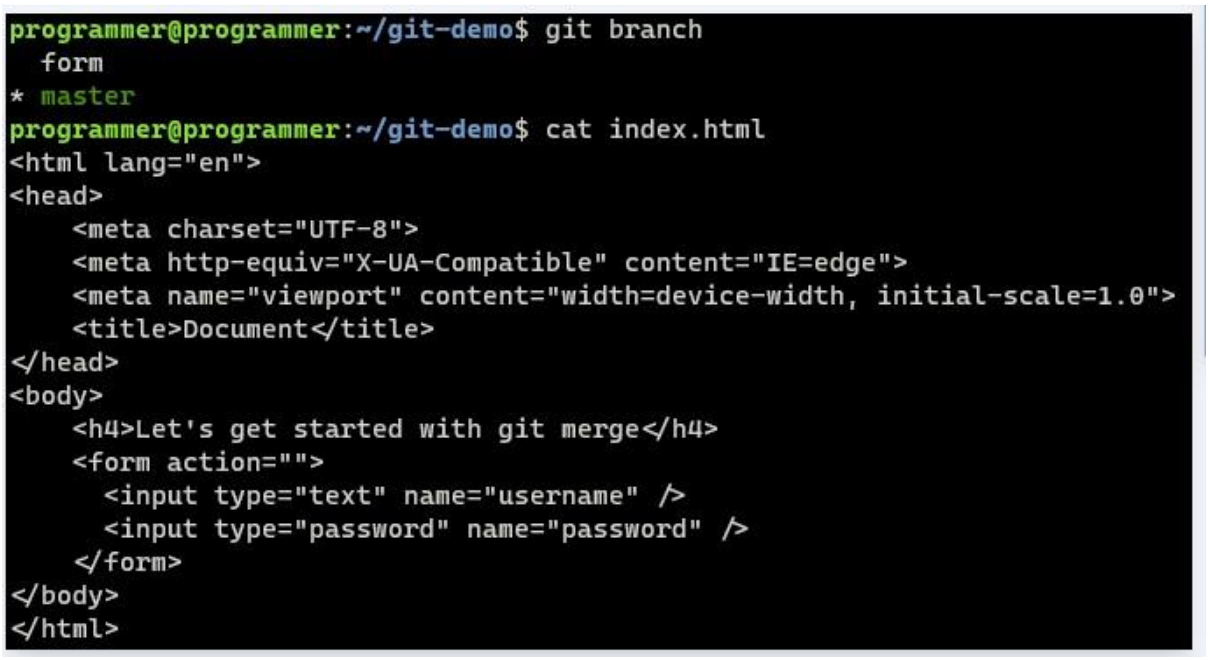


```
programmer@programmer:~/git-demo$ git checkout master
Switched to branch 'master'
programmer@programmer:~/git-demo$ git merge form
Updating 50f9ddd..253d1ed
Fast-forward
 index.html | 4 ++++
 1 file changed, 4 insertions(+)
```

3. Verify that the changes have been merged by verifying you are on the `master` branch and that the content has changed.



```
# Verify Branch
git branch
# View output
cat index.html
```



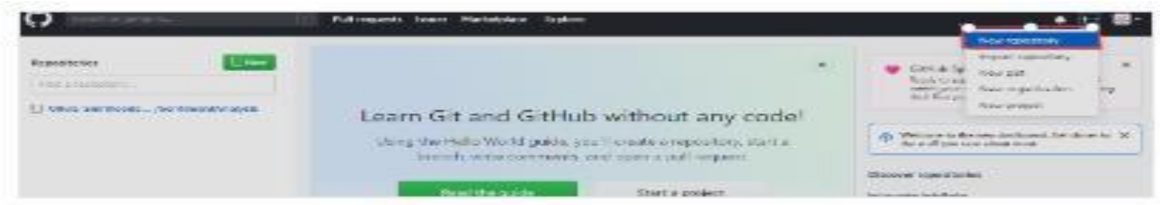
```
programmer@programmer:~/git-demo$ git branch
  form
* master
programmer@programmer:~/git-demo$ cat index.html
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h4>Let's get started with git merge</h4>
    <form action="">
      <input type="text" name="username" />
      <input type="password" name="password" />
    </form>
</body>
</html>
```

Push

Using Command line to PUSH to GitHub

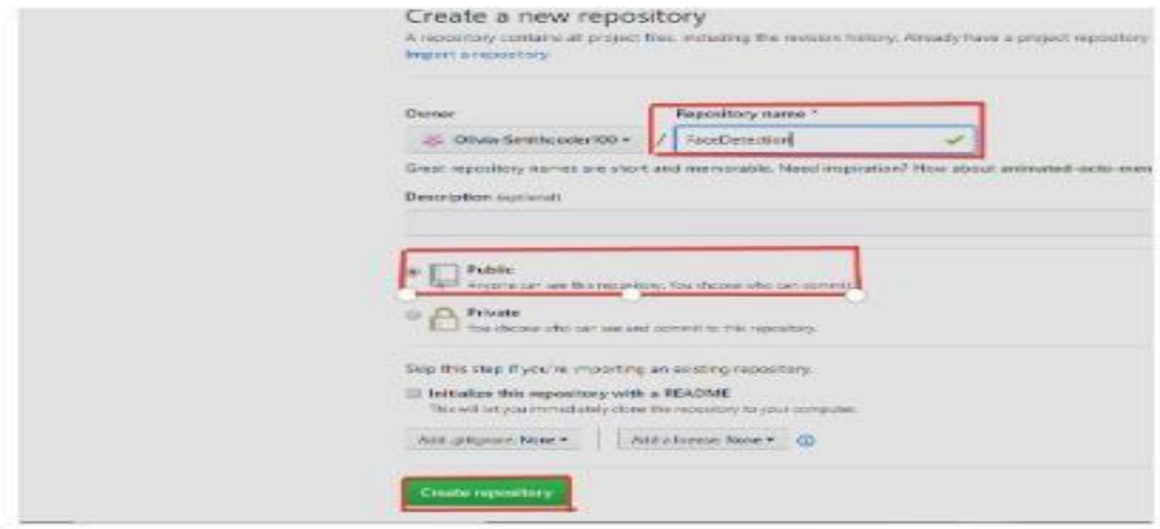
1. Creating a new repository

- 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.



Learn Git and GitHub without any code!

Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.


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-mem

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

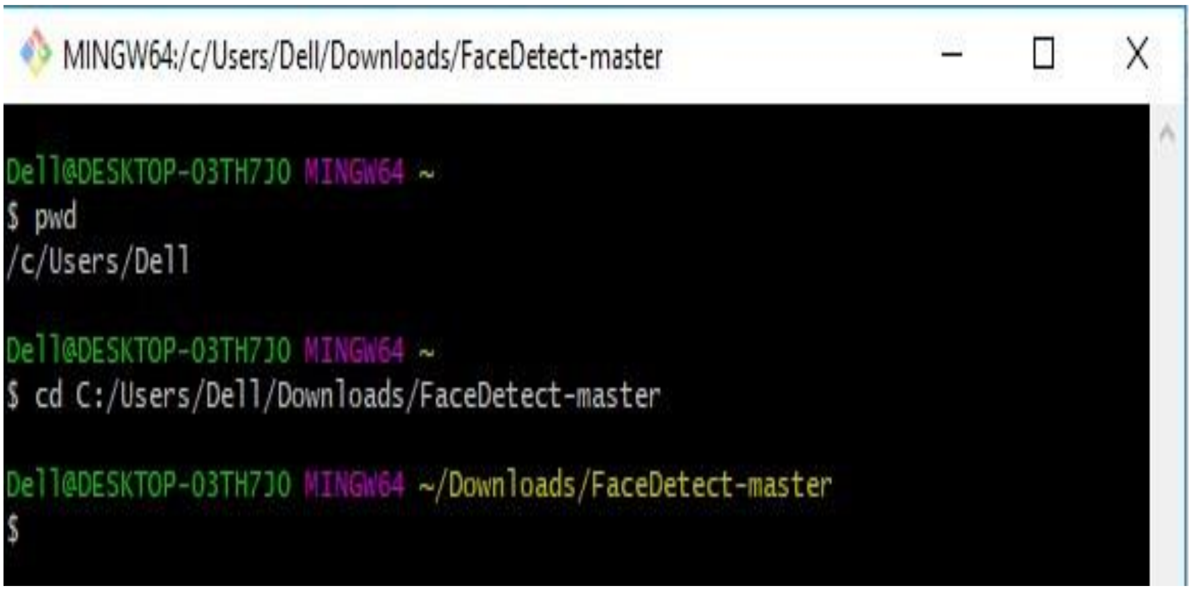
2. Open your Git Bash

- 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

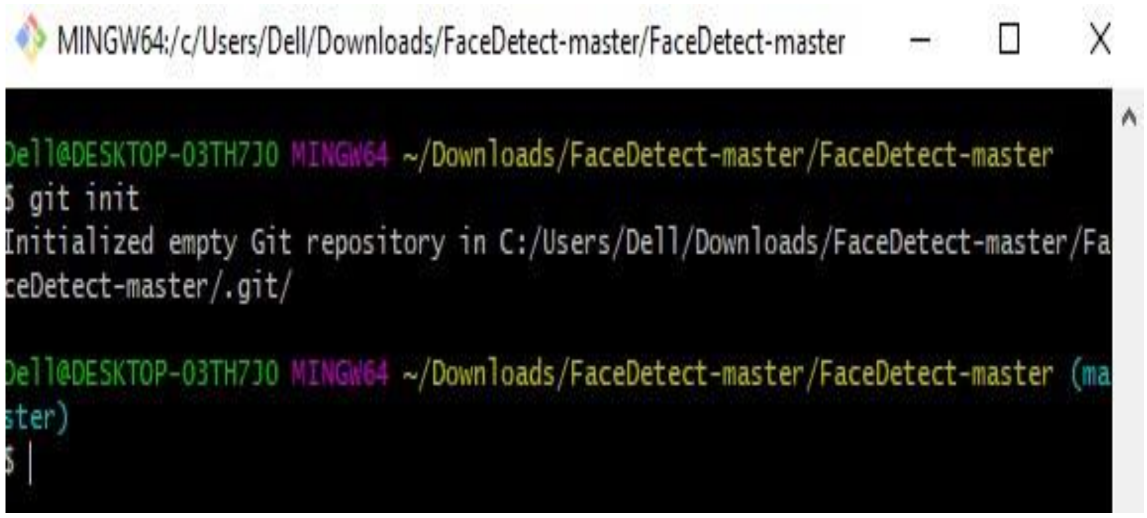
  file, `'path_name'`, i.e., `C:/Users/Dell/Downloads/FaceDetect-master` needs to be given. This command can identify the required file that you are looking to work with.



```
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

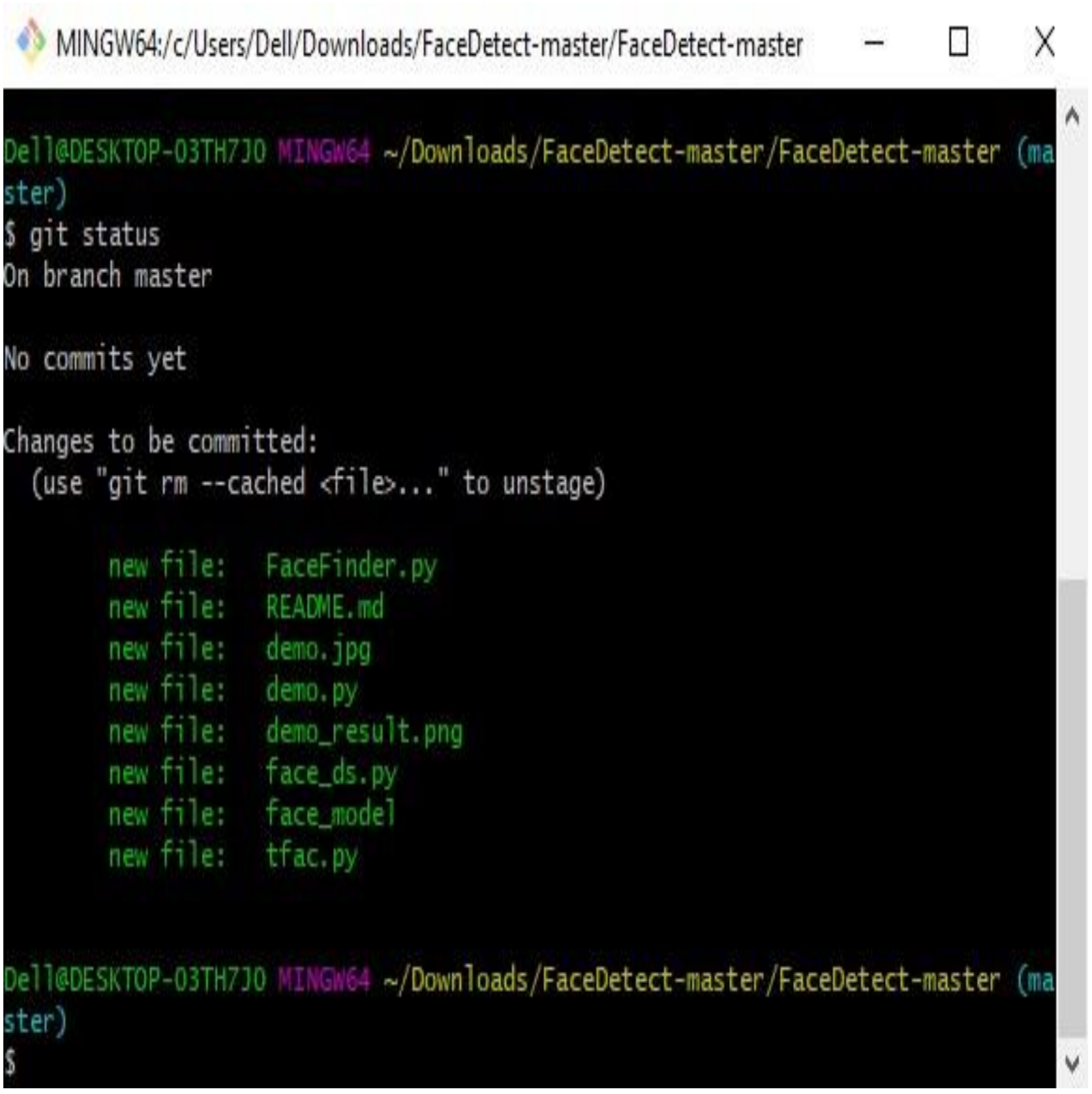
- Use `git init` to initialize the repository. It is used to create a new empty repository or directory consisting of files' with the hidden directory. '.git' is created at the top level of your project, which places all of the revision information in one place.



```
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/Fa
ceDetect-master/.git/

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

5. Add the file to the new local repository

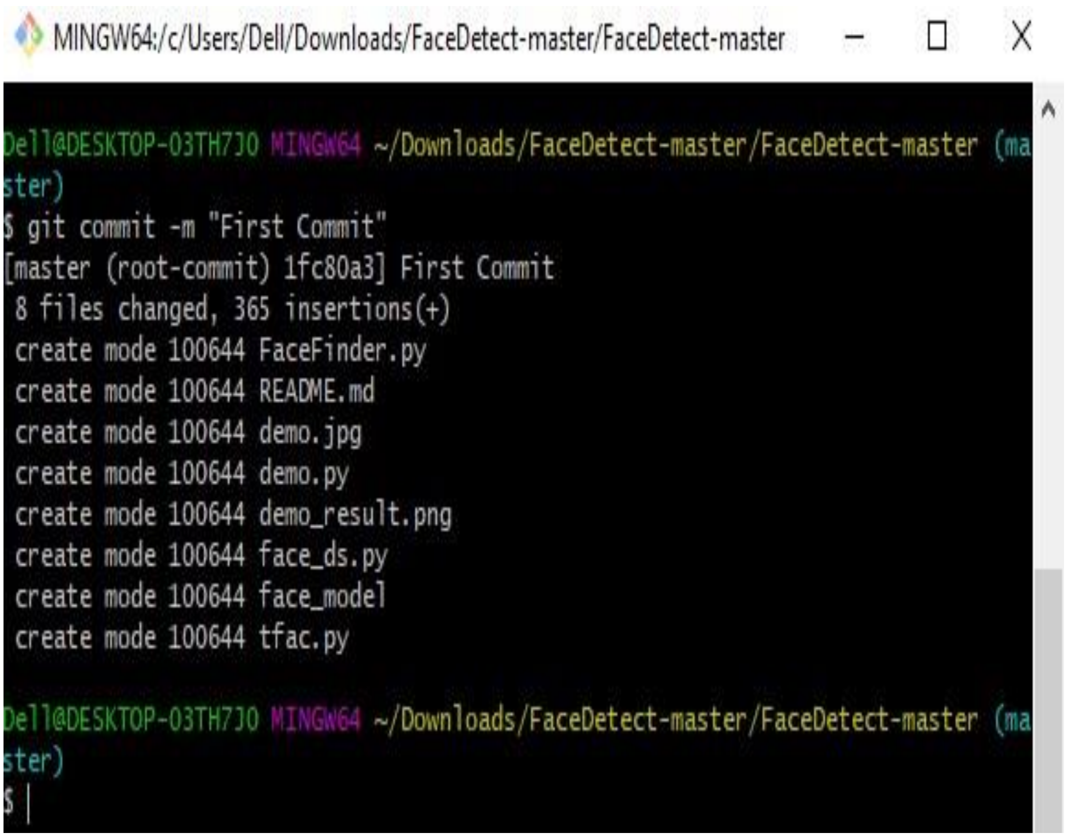
- Use `git add .` in your bash to add all the files to the given folder.
- Use `git status` in your bash to view all the files which are going to be staged to the first commit.



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

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

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

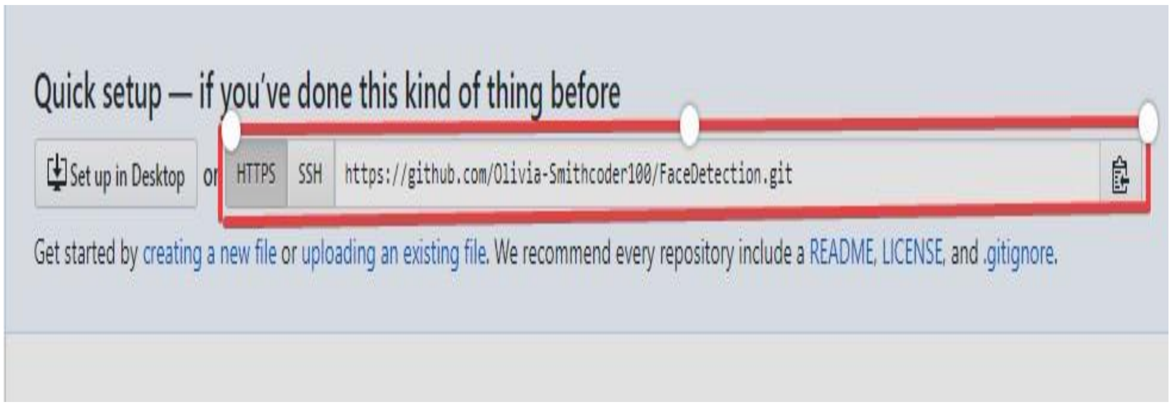
- You can create a commit message by `git commit -m 'your message'`, which adds the change to the local repository.
- `git commit` uses '-m' as a flag for a message to set the commits with the content where the full description is included, and a message is written in an imperative sentence up to 50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

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.



Quick setup — if you've done this kind of thing before

https://github.com/Olivia-Smithcoder100/FaceDetection.git

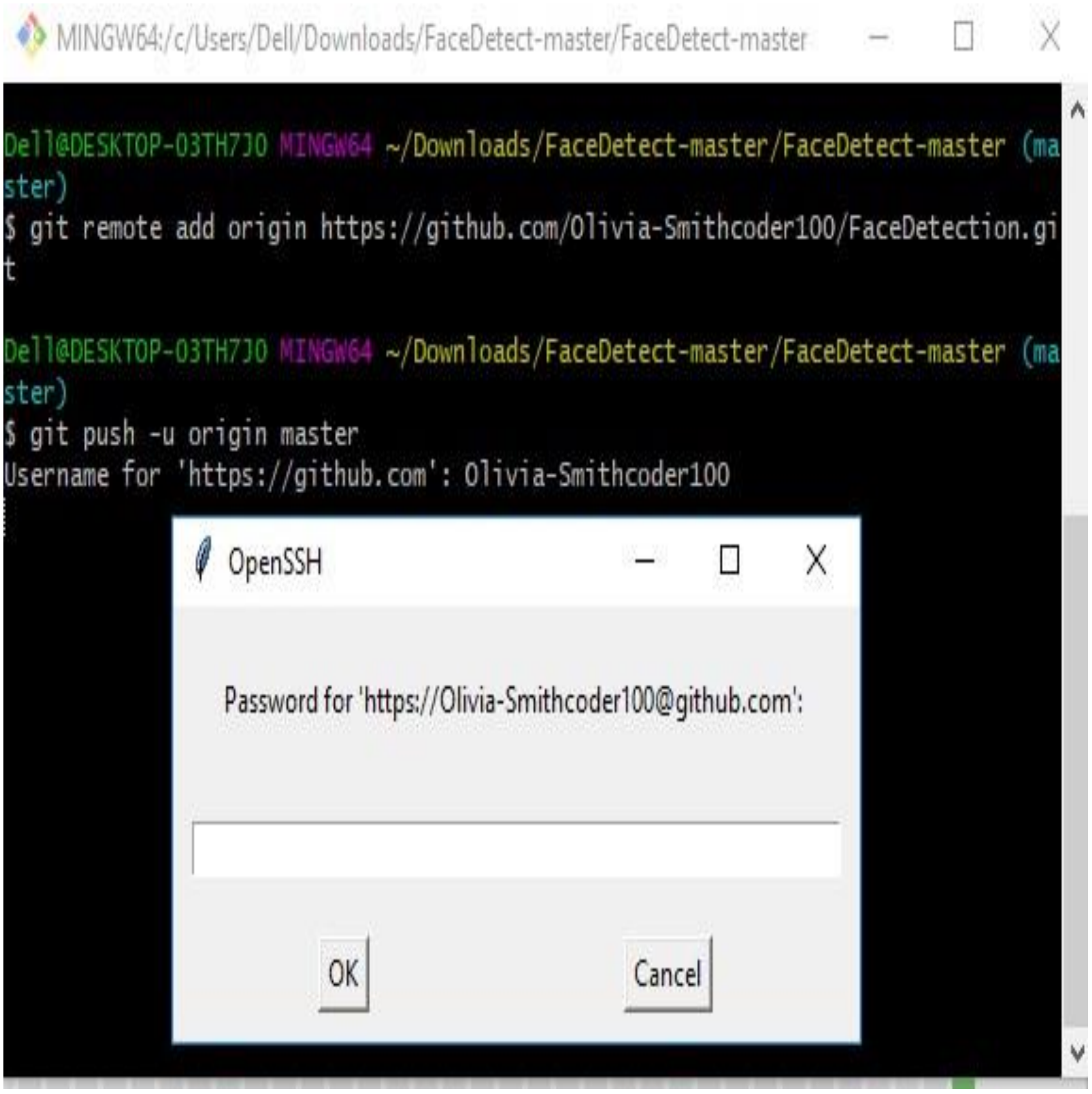
Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

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.



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

Dell@DESKTOP-03TH7J0 MINGW64 ~/Downloads/FaceDetect-master/FaceDetect-master (ma
ster)
$ git remote add origin https://github.com/Olivia-Smithcoder100/FaceDetection.gi
t

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

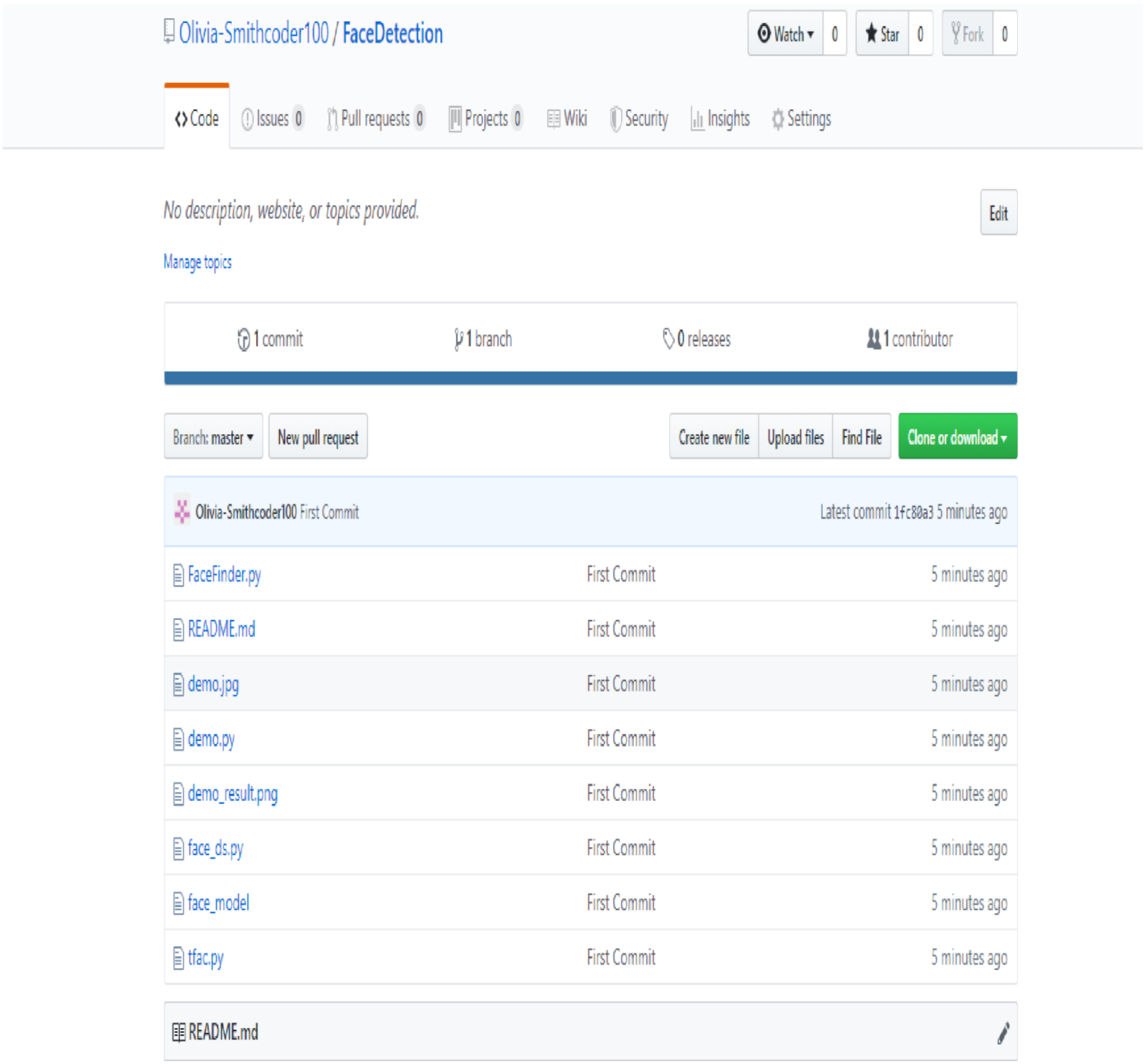
OpenSSH

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

10. View your files in your repository hosted on GitHub You can finally see the file

    hosted on GitHub.

Using GitHub Desktop to PUSH to your local content to GitHub.


Olivia-Smithcoder100 / FaceDetection

Watch 0 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Security Insights Settings

No description, website, or topics provided.

Edit

Manage topics

1 commit 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find File Clone or download

Olivia-Smithcoder100 First Commit Latest commit 1fc80a3 5 minutes ago

FaceFinder.py	First Commit	5 minutes ago
README.md	First Commit	5 minutes ago
demo.jpg	First Commit	5 minutes ago
demo.py	First Commit	5 minutes ago
demo_result.png	First Commit	5 minutes ago
face_ds.py	First Commit	5 minutes ago
face_model	First Commit	5 minutes ago
tfac.py	First Commit	5 minutes ago

README.md

GitHub Desktop is available to download for any operating system, and it gives the GUI(Graphical User Interface) platform to push your local content from your local repository to a remote repository like GitHub.

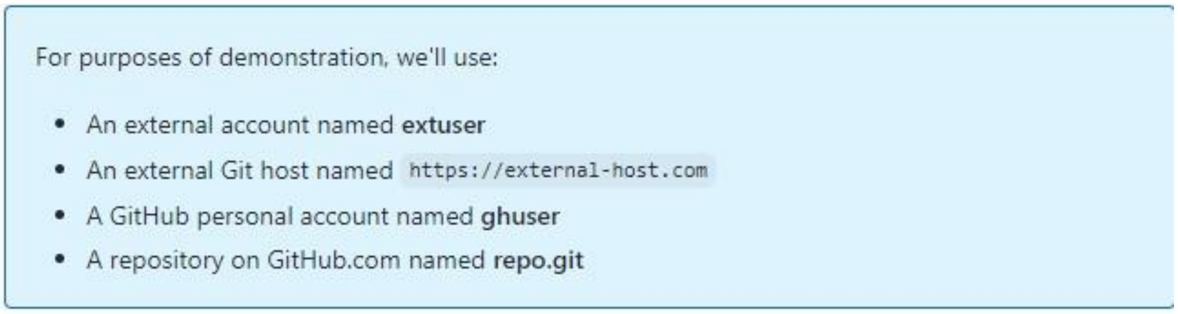
You need to open your GitHub account in your browser and the process of creating a new repository, i.e., step 1 is the same as mentioned above in "Using Command line to PUSH to GitHub".

pull(Importing) a Git repository using the command line

If GitHub Importer is not suitable for your purposes, such as if your existing code is hosted on a private network, then we recommend importing using the command line.

Before you start, make sure you know:

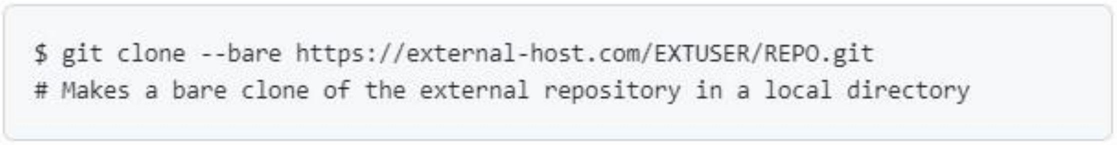
- Your GitHub username
- The clone URL for the external repository, such as `https://externalhost.com/user/repo.git` or `git://external-host.com/user/repo.git` (perhaps with a `user@` in front of the `external-host.com` domain name)



For purposes of demonstration, we'll use:

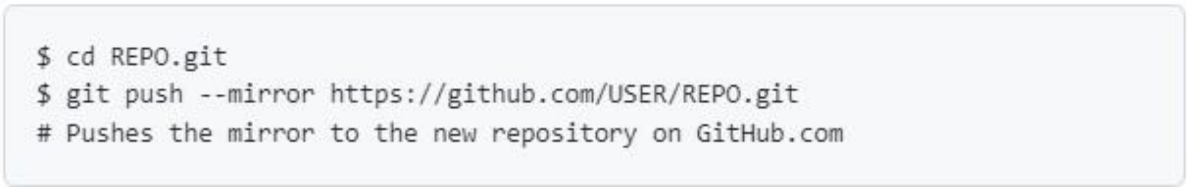
- An external account named **extuser**
- An external Git host named `https://external-host.com`
- A GitHub personal account named **ghuser**
- A repository on GitHub.com named **repo.git**

1. Create a new repository on GitHub. You'll import your external Git repository to this new repository.
2. On the command line, make a "bare" clone of the repository using the external clone URL. This creates a full copy of the data, but without a working directory for editing files, and ensures a clean, fresh export of all the old data.



```
$ git clone --bare https://external-host.com/EXTUSER/REPO.git
# Makes a bare clone of the external repository in a local directory
```

3.Push the locally cloned repository to GitHub using the "mirror" option, which ensures that all references, such as branches and tags, are copied to the imported repository.



```
$ cd REPO.git
$ git push --mirror https://github.com/USER/REPO.git
# Pushes the mirror to the new repository on GitHub.com
```

4.Remove the temporary local repository.

```
$ cd ..
$ rm -rf REPO.git
```