GIT and GITHUB

Created By-
Shridhar Shende

What is Git?

Git is a version control system that lets you track changes you make to your project's files over time. With Git, you can revert to various states of your files.

What is GitHub?

GitHub is an online hosting service for Git repositories.

GitHub lets you store your repo on their platform. Another awesome feature that comes with GitHub is the ability to collaborate with other developers from any location.

Git installation and configuration.

download the latest version from the official website, https://git-scm.com/ and install.

How to config git

Step 1: create an account on github.com and get user name & email id

Step 2: open git bash terminal

Step 3: To set username type and execute a command

```
git config --global user.name "YOUR_USERNAME"
```

Step 4: To set email type and execute a command

```
git config --global user.email "YOUR_EMAIL"
```

How to Create and Initialize a Project in Git

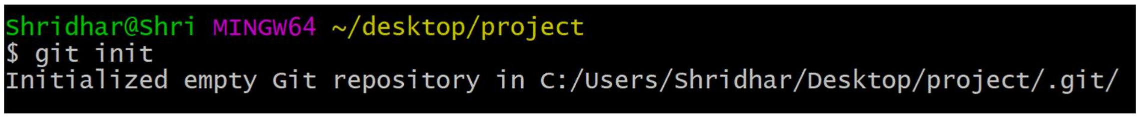
Step 1: Create a folder called project

Step 2: change the project file path on the git bash terminal

Use command: `cd project_file_path`

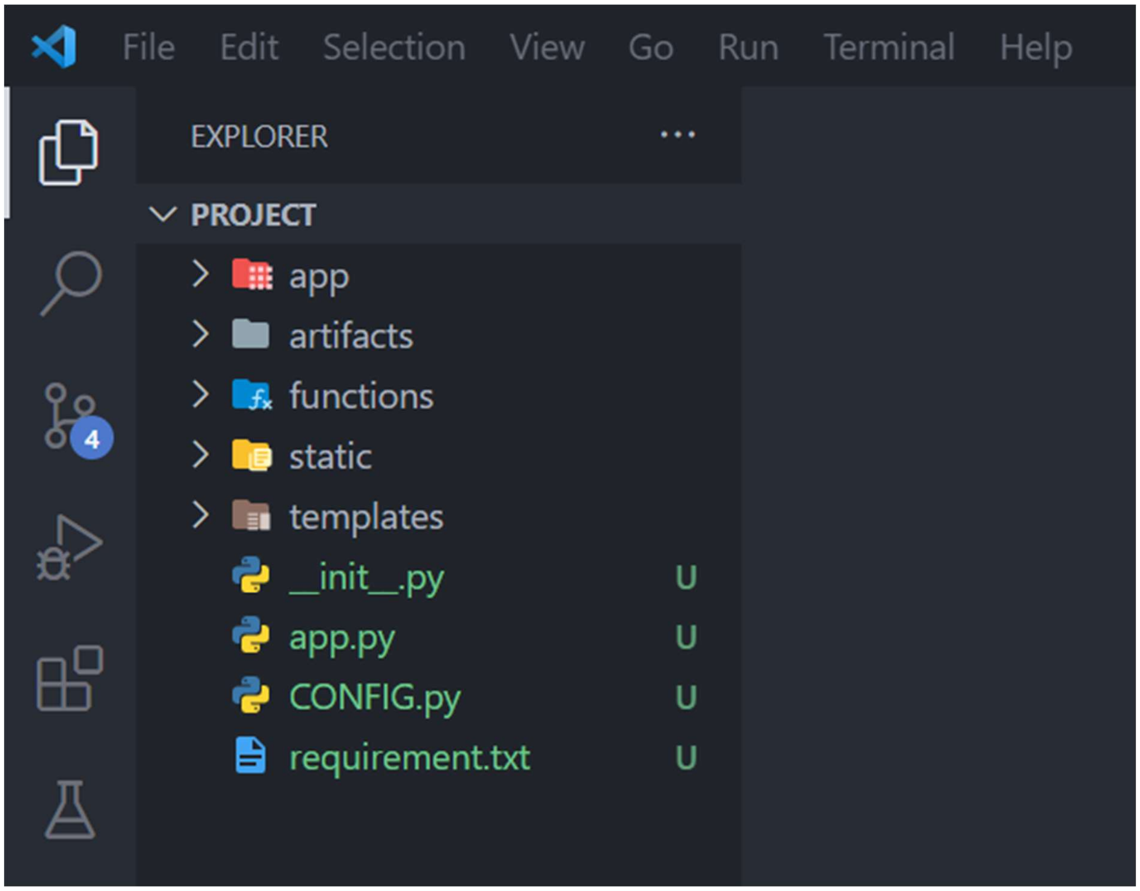
Step 3: Now to initialize your project, simply run

`git init`



```
Shridhar@Shri MINGW64 ~/desktop/project
$ git init
Initialized empty Git repository in C:/Users/Shridhar/Desktop/project/.git/
```

Step 4: Assume your project files structure will look like this

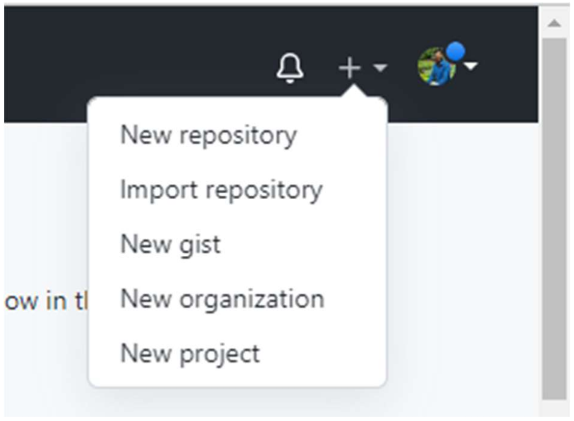


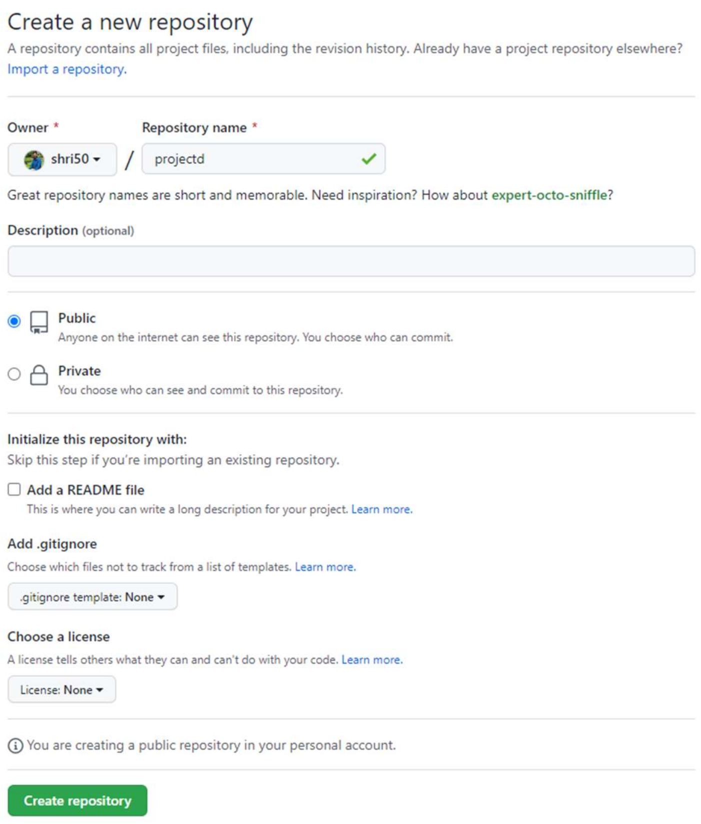
Push a repository to GitHub

Step 1 – Login to GitHub account

Step 2 – Create a repository

- click on the + symbol on the top right corner
- choose "New repository".
- Give your repo a name.
- scroll down and click on "Create repository".





Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner * shri50 / Repository name * projectd

Great repository names are short and memorable. Need inspiration? How about expert-octo-sniffle?

Description (optional)

(●) Public
Anyone on the internet can see this repository. You choose who can commit.

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

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

☐ Add a README file
This is where you can write a long description for your project. Learn more.

Add .gitignore
Choose which files not to track from a list of templates. Learn more.
.gitignore template: None

Choose a license
A license tells others what they can and can't do with your code. Learn more.
License: None

You are creating a public repository in your personal account.

Create repository

Step 3 – Add and commit file(s)

Before we "add" and "commit" let's understand the stages of a file being tracked by Git.

1. Modified state
2. Staged state
3. Committed state

Modified state

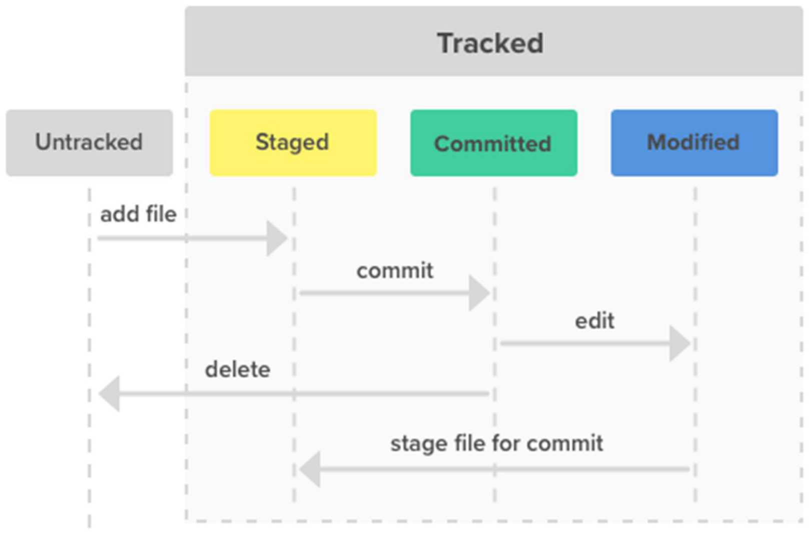
this state has some changes made to it but it's not yet saved. This means that the state of the file has been altered/changed from its previous state.

Staged state

staged state means it is ready to be committed.
In this state, all necessary changes have been made so the next step is to move the file to the commit state

Committed state

committed state when all the changes made to the file have been saved in the local repo. Files in the committed stage are files ready to be pushed to the remote repo (on GitHub).



How to add files in Git

`git add file_name`

>> for a single file

`git add .`

>> for all files.

`git add -all`

>> for all files.

e.g

`git add app.py`

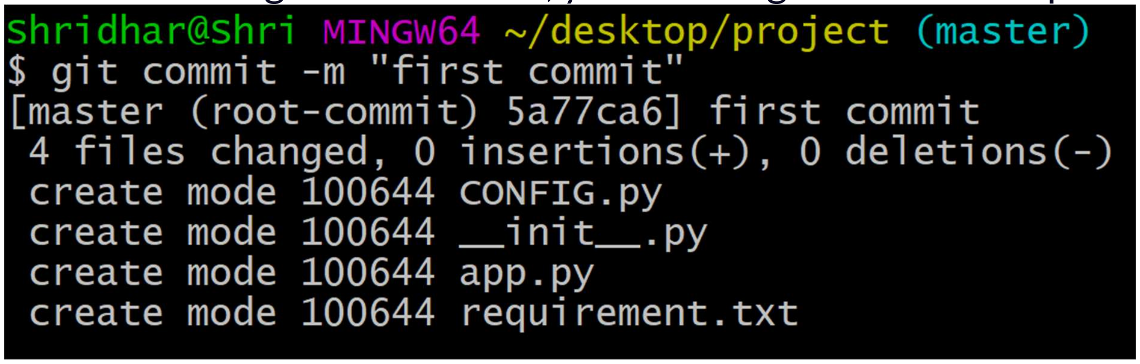
How to commit files in Git

`git commit -m "first commit"`

`git commit` tells Git that all the files staged are ready to be committed.

`-m "first commit"` is the commit message.

After executing this command, you should get a similar response:



```
Shridhar@Shri MINGW64 ~/desktop/project (master)
$ git commit -m "first commit"
[master (root-commit) 5a77ca6] first commit
 4 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 CONFIG.py
 create mode 100644 __init__.py
 create mode 100644 app.py
 create mode 100644 requirement.txt
```

Step 4 – Push the repository to GitHub

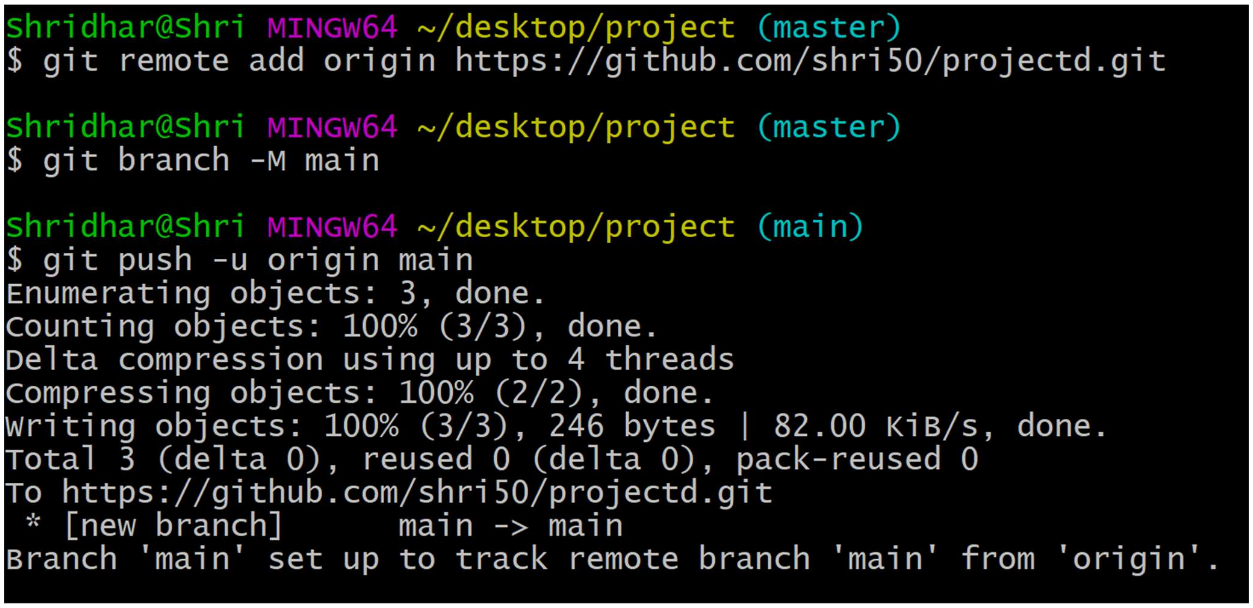
```
git remote add origin https://github.com/shri50/projectd.git
git branch -M main
git push -u origin main
```

The first command `git remote add origin https://github.com/shri50/projectd.git` creates a connection between your local repo and the remote repo on Github.

The second command `git branch -M main` changes your main branch's name to "main".
The default branch might be created as "master", but "main" is the standard name for this repo now

Last command `git push -u origin main` pushes your repo from your local device to GitHub

You should get a response similar as follow:



```
Shridhar@Shri MINGW64 ~/desktop/project (master)
$ git remote add origin https://github.com/shri50/projectd.git

Shridhar@Shri MINGW64 ~/desktop/project (master)
$ git branch -M main

Shridhar@Shri MINGW64 ~/desktop/project (main)
$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 246 bytes | 82.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/shri50/projectd.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

How to Use Branches in Git

branches, you can create a copy of a file you would like to work on without messing up the original copy.

you can either merge these changes to the original copy or just let the branch remain independent

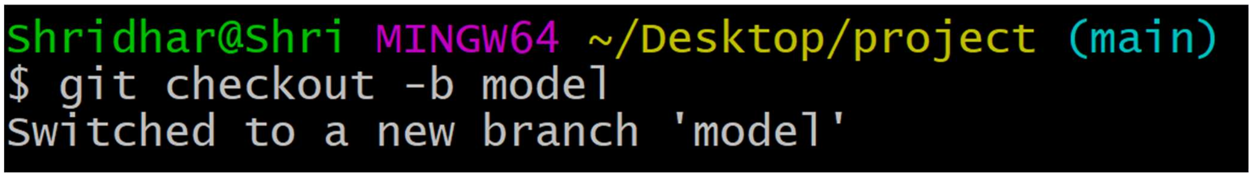
To create a new branch, run this command:

`git checkout -b model.`

`checkout` tells Git it is supposed to switch to a new branch

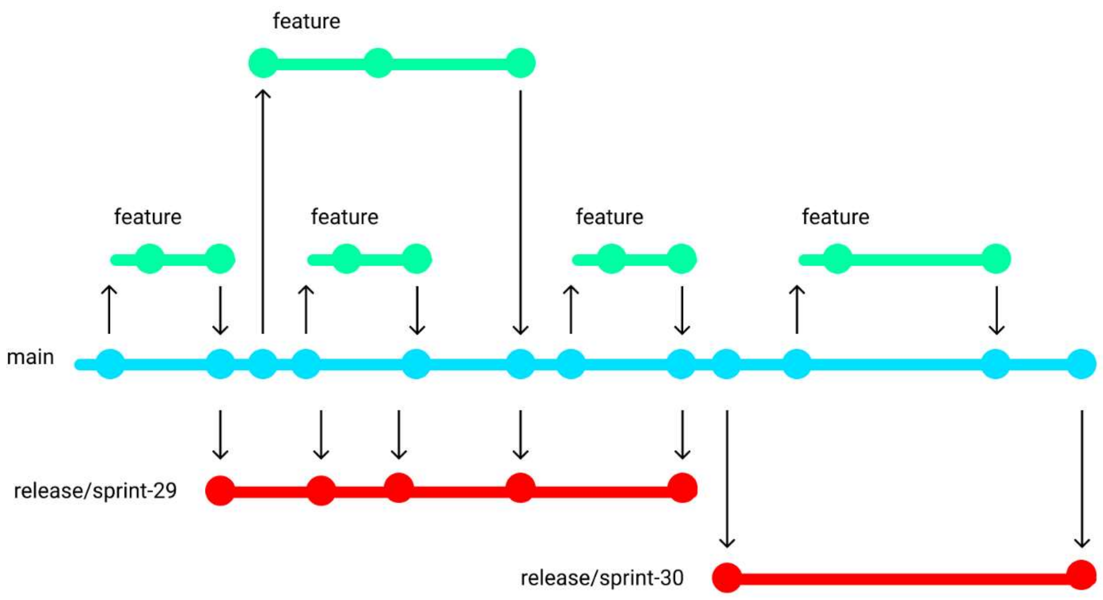
`-b` tells Git to create a new branch.

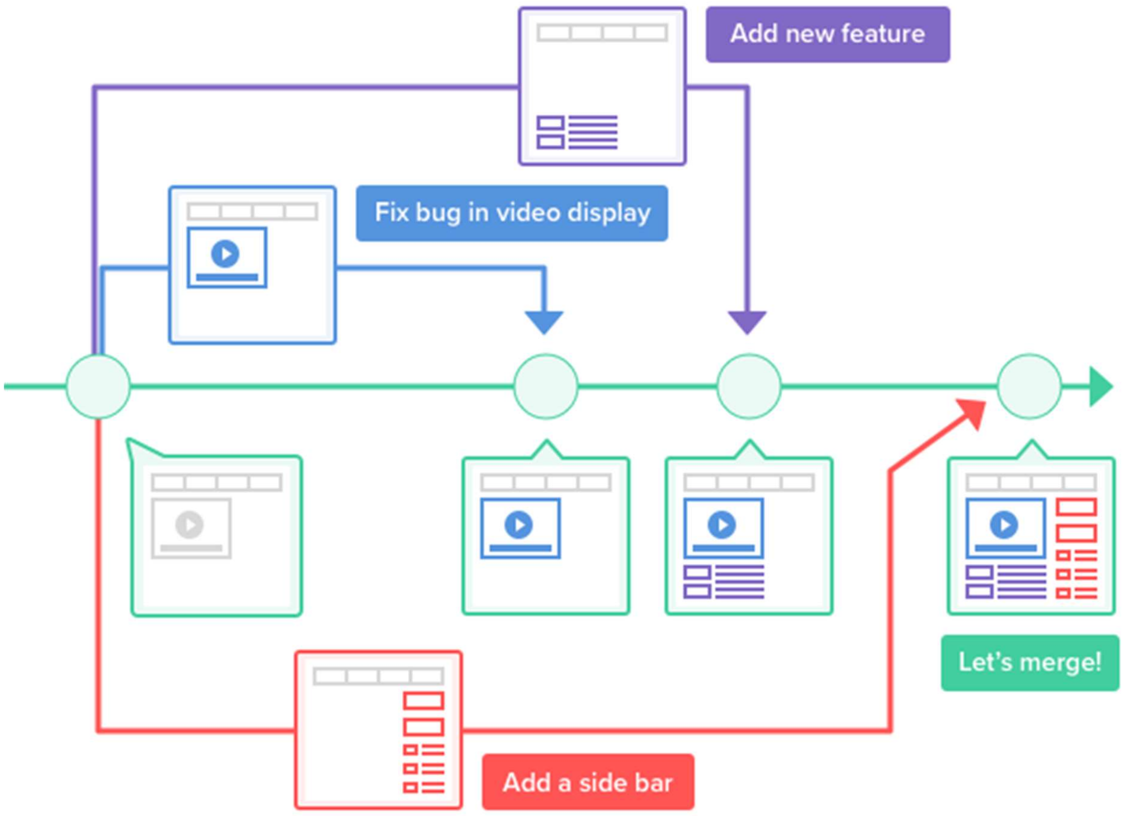
`model` is the name of the branch to be created and switched to.



```
Shridhar@Shri MINGW64 ~/Desktop/project (main)
$ git checkout -b model
Switched to a new branch 'model'
```

Graphical representation of How branch works in git

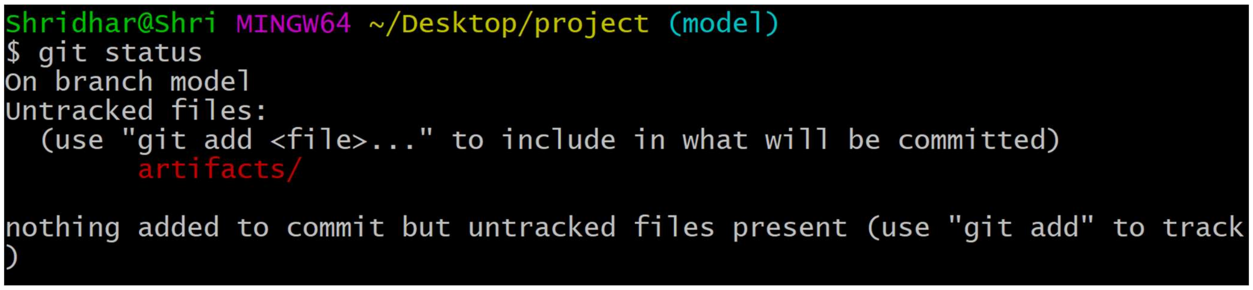




We created the new branch from the state of our last commit.

Let's now add more ml models to this new branch.

Again we have our files in the untracked stage. [check using `git status`]

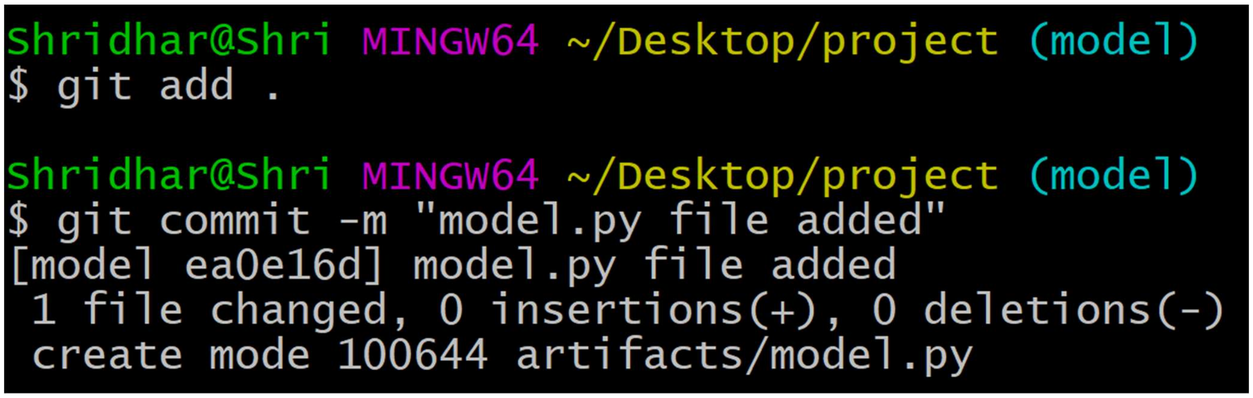


```
Shridhar@Shri MINGW64 ~/Desktop/project (model)
$ git status
On branch model
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        artifacts/

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

You know the process to bring your changes to the committed state

1. `git add .`
2. `git commit -m “your msg”`



```
Shridhar@Shri MINGW64 ~/Desktop/project (model)
$ git add .

Shridhar@Shri MINGW64 ~/Desktop/project (model)
$ git commit -m "model.py file added"
[model ea0e16d] model.py file added
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 artifacts/model.py
```

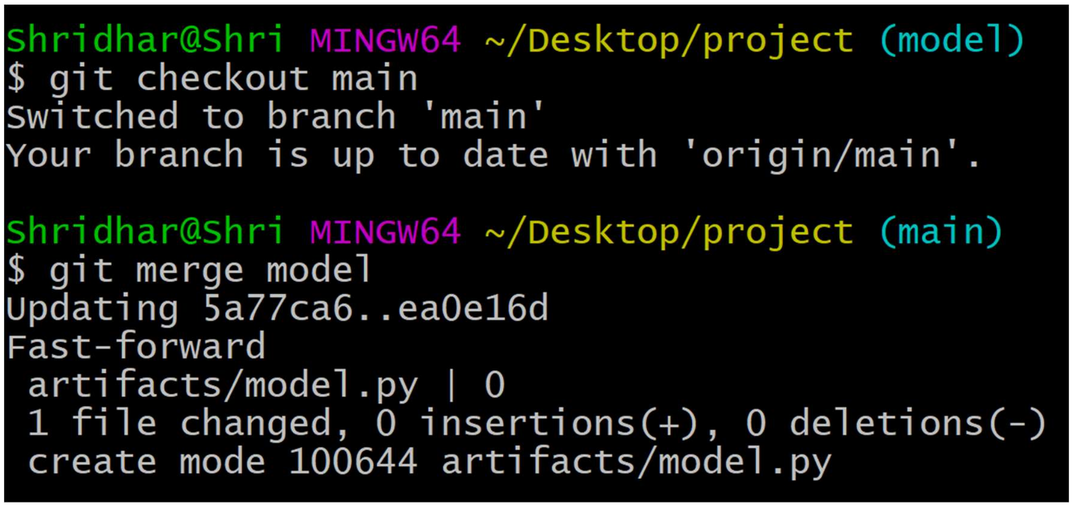
After committing your model branch,

switch back to the main branch by running this command:

`git checkout main`

Now we can merge the changes we made in the model branch into the main branch by running command

`git checkout model`



```
Shridhar@Shri MINGW64 ~/Desktop/project (model)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

Shridhar@Shri MINGW64 ~/Desktop/project (main)
$ git merge model
Updating 5a77ca6..ea0e16d
Fast-forward
 artifacts/model.py | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 artifacts/model.py
```

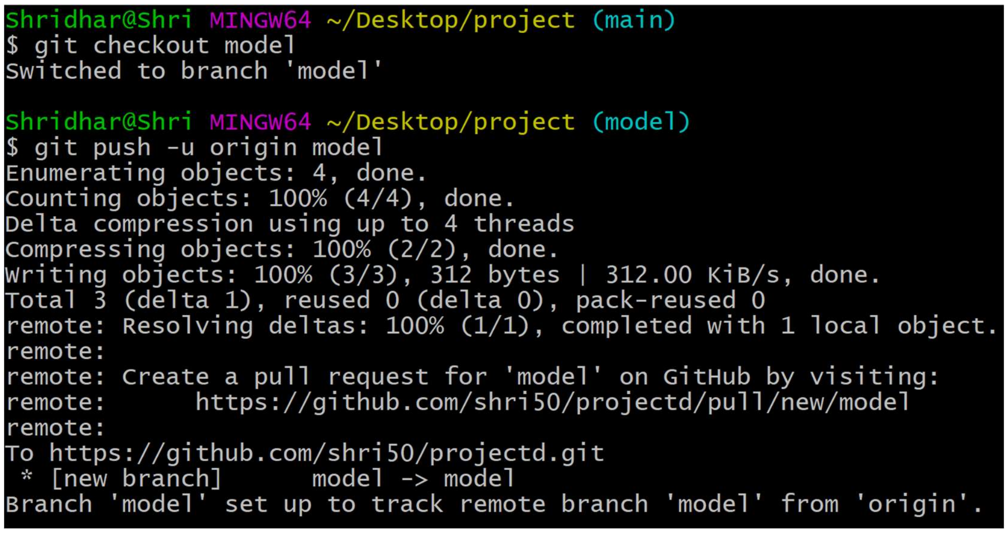
How to Push branch to Repository in Git

If you push the main branch to GitHub repo [git push -u origin main] . you'll see that the model branch will not be pushed to the Github repo.

It will only remain in your local repo.

If you would like to push your model branch:

- `git checkout model`
- `git push -u origin model`



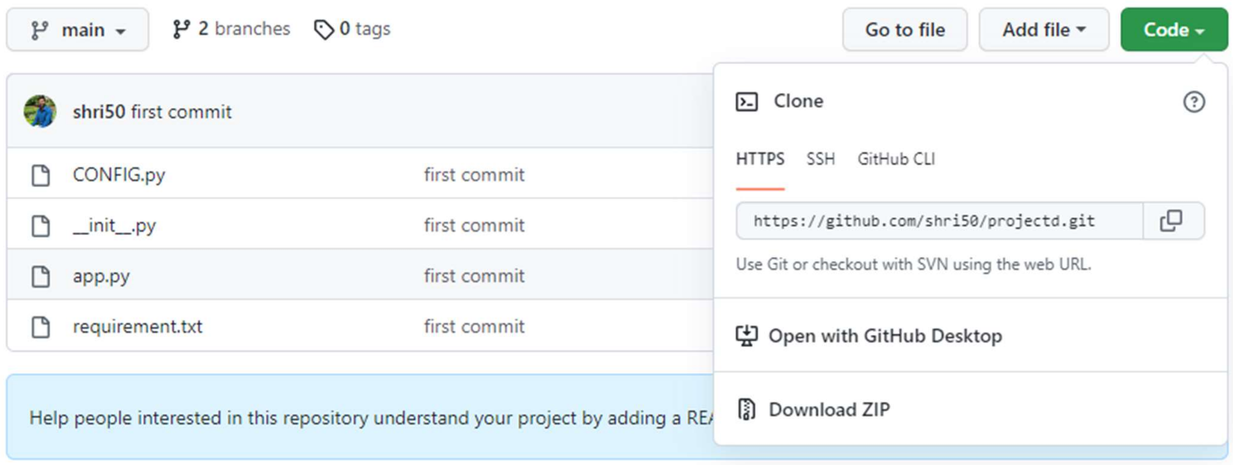
```
Shridhar@Shri MINGW64 ~/Desktop/project (main)
$ git checkout model
Switched to branch 'model'

Shridhar@Shri MINGW64 ~/Desktop/project (model)
$ git push -u origin model
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 312 bytes | 312.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'model' on GitHub by visiting:
remote:      https://github.com/shri50/projectd/pull/new/model
remote:
To https://github.com/shri50/projectd.git
 * [new branch]      model -> model
Branch 'model' set up to track remote branch 'model' from 'origin'.
```

How to Pull a Repository in Git

to pull in Git means to clone a remote repository's current state into your computer/repository.

Step 1: Goto GitHub, and on your repository's main page you should see a green button that says "Code". Click on the button

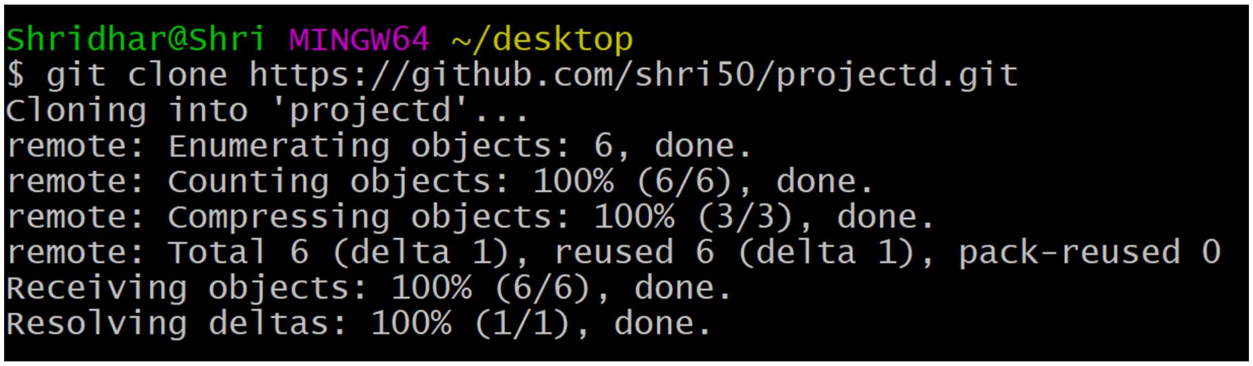


Step 2: Go on and copy the HTTPS URL.

Step 3: Open git bash terminal

Step 4: Select the path where you want to clone repo.

Step 3: Run command : `git clone HTTPS_URL`



```
Shridhar@Shri MINGW64 ~/desktop
$ git clone https://github.com/shri50/projectd.git
Cloning into 'projectd'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 6 (delta 1), reused 6 (delta 1), pack-reused 0
Receiving objects: 100% (6/6), done.
Resolving deltas: 100% (1/1), done.
```

How to Pull changes in Local repo

Some other teams made changes are merged into the main branch on the Github repo.

Now you also want your local repo to be updated with recent changes. Or we can say the latest committed state

We can achieve this using command:

```
git pull
```

How to Delete Branch

Delete from local repository

```
    git branch -d branch_name
```

Delete from remote repository

```
    git push origin –delete branch_name
```