

Git and GitHub

Introduction ::

Git is a version control software used to manage projects amongst different versions and people. It makes life easier by allowing you and your team to look at code, contribute to it and review other people's work.

Jargons :

- Repositories :
 - These are like folders containing your projects.
 - You have different repositories for different projects that you work on or contribute to.
 - These are of two types :
 - Local : Stored on your machine
 - Remote : Stored on the git client.
 - Repositories that are shared can be :
 - Public : These are open-source. Shared with everyone. Anyone can view/save the code and everything that the repo contains.
 - Private : These are shared with people that you choose. It may be within organization and/or outside.
- Branches :
 - Let's say you and your teammates are working on the same project but on different parts of the code.
 - It gets very difficult to sync all the changes by sharing files over WhatsApp.

- A better way to work among different features for the same repository is by creating branches. These are another line of development. The default branch containing the final code is called the 'main' branch.
- A branch is created to work on a specific feature and you essentially have a copy of the main branch. You add your code to this branch and you save your changes by making 'commits'.
- Commits :
 - Commits are like saving your code.
 - Your most recent commit holds the current state of the repository.
 - You can think of it as a linked list. Where every new commit is appended to the current list and the HEAD pointer is updated to the last element.
 - For any given commit you can go back or 'rollback' to a previous commit by looking at it's 'PARENT' pointer. This is useful if you had your code working and then made some changes after which your code is failing. You can just rollback to a commit.
 - If a commit has two PARENT pointers, it is a result of a merge commit where two branches were merged into one.
 - It is recommended that you create commits regularly and after every major change or bug fix in your code.
 - It is also recommended you keep different additions to a code in separate commits and not mix different feature changes into one commit for better rollback options.
- Pull Requests :
 - So, you're done with all your changes and have implemented a new feature in your project. The thing to note is that these changes were made in a separate branch and production always looks at the main branch of the code.
 - How do you get your new and fresh code to the main branch ? This is where Pull Requests come in.
 - You basically say, "Hey I am done with my feature. Can you come look at the changes and add my code to the main branch ?" Then people review your code and if it gets approved, it is 'MERGED' to the main branch.
 - The feature branch is usually deleted after this.

- Clone :
 - Okay so you have a repository over on GitHub. But how do you contribute to it. You need the code on your laptop, right ?
 - So, we use the clone command in any terminal. (Provided that you have git installed). This creates a local instance of the repository and you can, now, make your desired changes/commits. These will be stored on your local machine.
 - You might need to set up some credentials to work with private repositories.
- Push and Pull :
 - You work on your local machine and are doing commits on it. But these changes are happening in your machine.
 - To get your changes to actually show up on the repository, you need the push command. This takes all the commits that you made and "uploads" it to your github repository.
 - Now your teammates need to see these changes. So, they use the pull command to get these changes to show up on their PC.
 - It is recommended that you do not alter the main branch with any commits and always work in branches.
 - It is also recommended to run the 'pull' before 'push' to sync everything to the server and then update your code to avoid merge conflicts.

Gitting Started ::

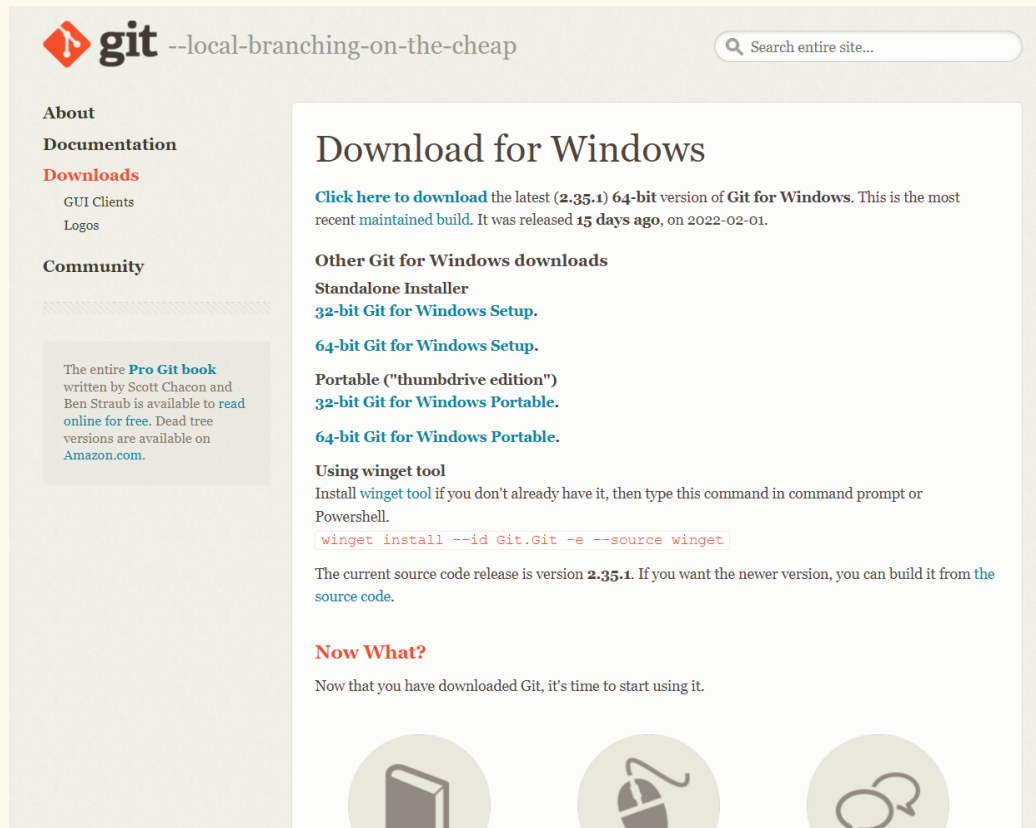
(Pun Intended)

What you need :

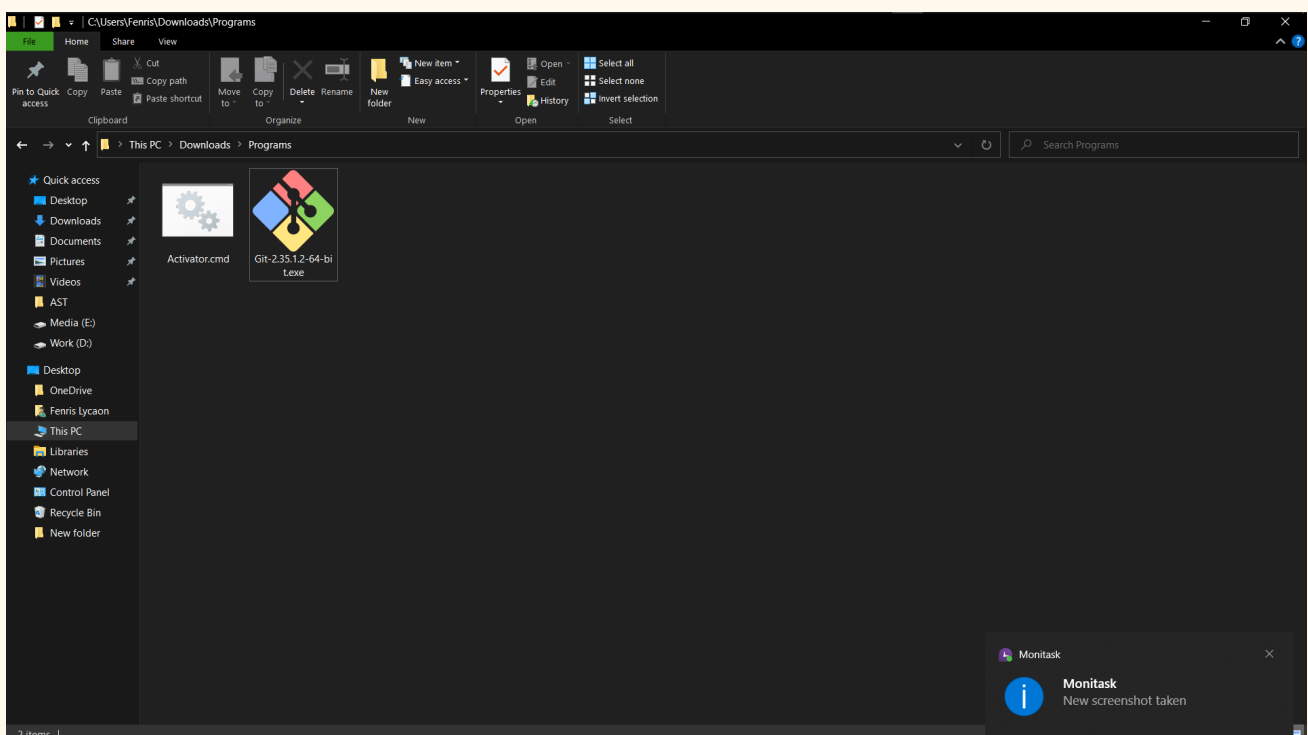
- Basic working knowledge of git commands like add, commit, pull, push and merge.
- CodeBase and a team to work with.
- An account over at GitHub/Gitlab/BitBucket. In DCS, we're working with GitHub.

I would recommend installing the [GIT SCM](#) tool for working with git on windows. On linux and macOS the respective git packages are enough but I'd still recommend installing github-cli for better integration.

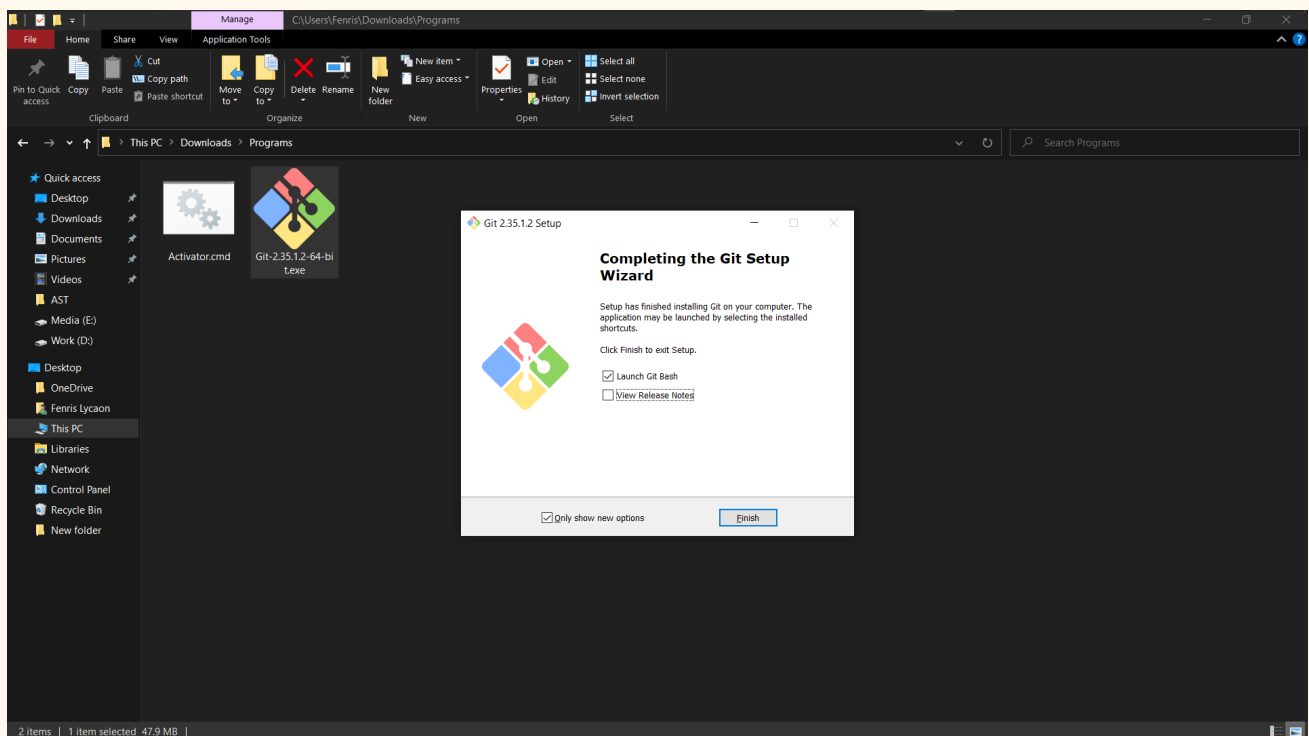
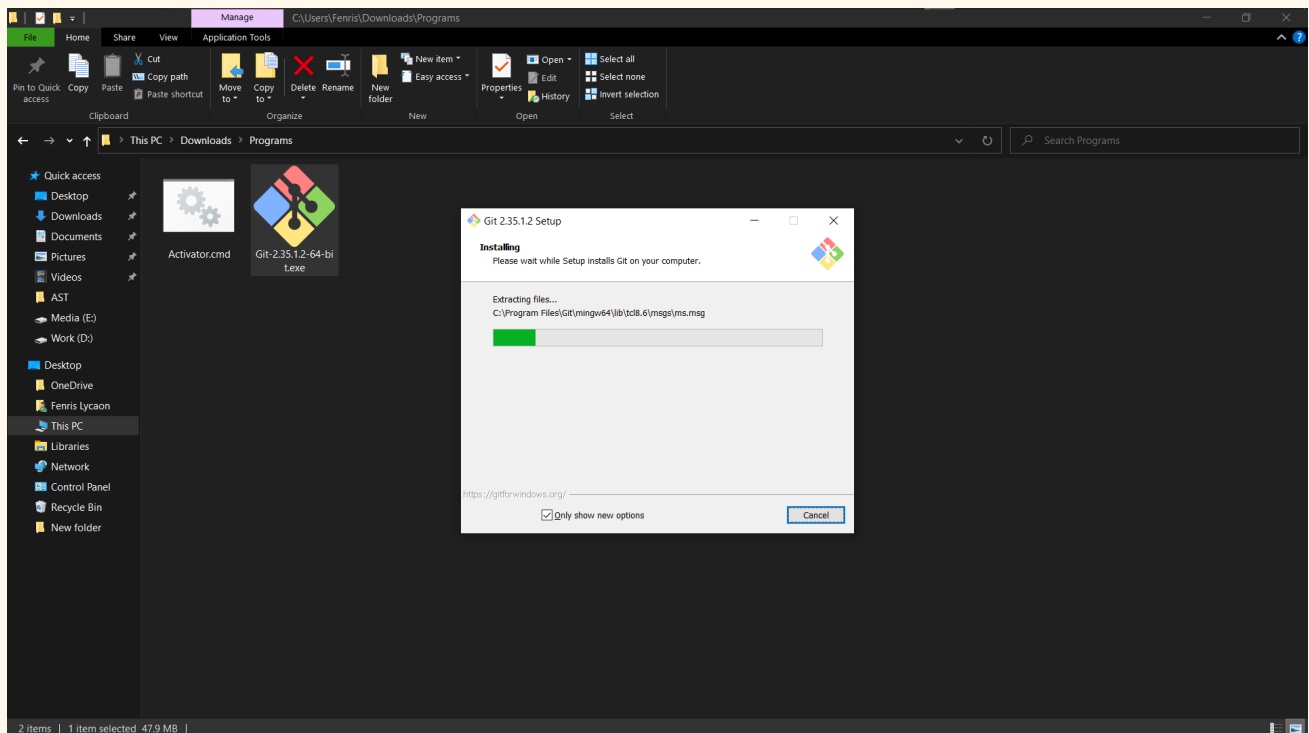
Download from here :



Installing the program :



It's going to take a while ::



- Once it's installed , Open git bash.

- Once you have downloaded and installed git, you need to provide your credentials to it so you can clone, commit and push on the remote repos. This can be done by the following command :

- `git config --global user.name "your username"`
 - `git config --global user.email "your_email_address@example.com"`

- To verify your configuration, run :

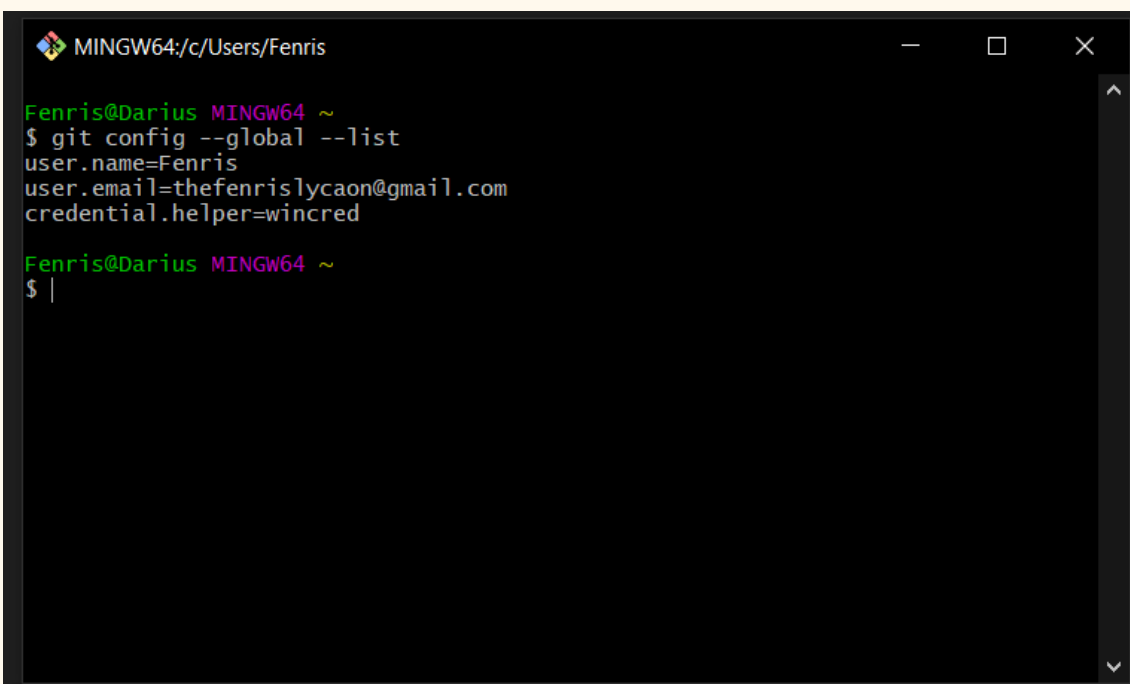
- `git config --global --list`

- To save your credentials, you can :

- `git config --global credential.helper store`

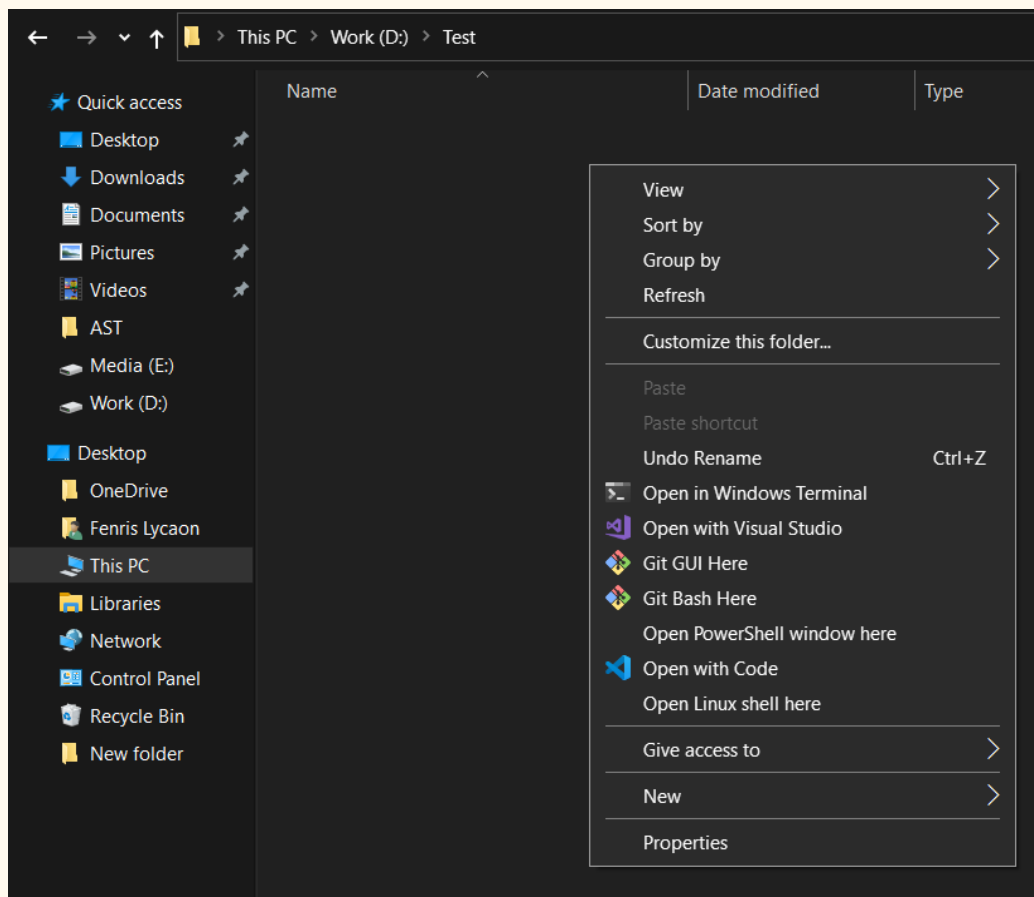


A terminal window titled "MINGW64:/c/Users/Fenris" with a dark background. The prompt "Fenris@Darius MINGW64 ~" is shown in green and pink. Below the prompt is a white dollar sign followed by a vertical bar, indicating a command prompt.



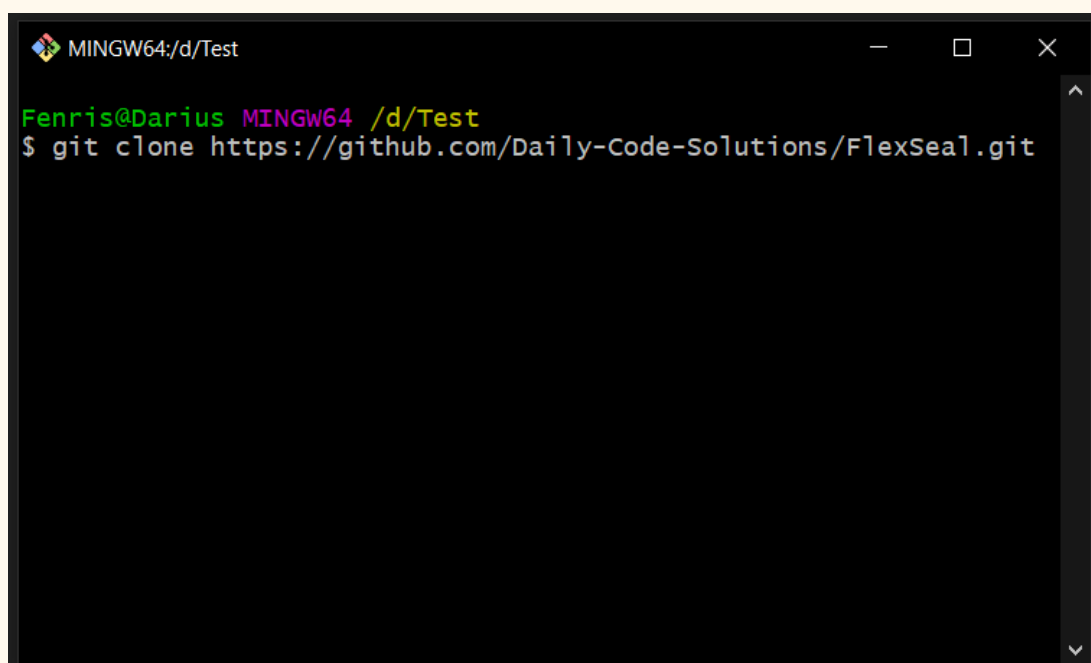
A terminal window titled "MINGW64:/c/Users/Fenris" with a dark background. The prompt "Fenris@Darius MINGW64 ~" is shown in green and pink. The command `$ git config --global --list` has been entered, and the output is displayed: `user.name=Fenris`, `user.email=thefenrislycaon@gmail.com`, and `credential.helper=wincred`. Below the output, the prompt is shown again with a white dollar sign followed by a vertical bar.

Now open the folder where you want to get your project repository. Right click and open a terminal. Any terminal should work if you have configured git properly and have the binaries in your PATH variable. Even if you mess up and git is not working on CMD or powershell, you can always rely on GIT BASH to work.



- Now to clone a repository you can :

○ **git clone <url>**



```
MINGW64:/d/Test
Fenris@Darius MINGW64 /d/Test
$ git clone https://github.com/Daily-Code-Solutions/FlexSeal.git
Cloning into 'FlexSeal'...
remote: Enumerating objects: 260, done.
remote: Counting objects: 100% (260/260), done.
remote: Compressing objects: 100% (138/138), done.
remote: Total 260 (delta 144), reused 219 (delta 106), pack-reuse
d 0
Receiving objects: 100% (260/260), 16.99 MiB | 2.02 MiB/s, done.
Resolving deltas: 100% (144/144), done.

Fenris@Darius MINGW64 /d/Test
$ |
```

Once the repository is cloned, you need to open a terminal into this new directory.

The very first step of contribution is creating a new branch. It is recommended that you create a new branch for every feature that is introduced.

- To create a new branch :
 - **git branch <branch-name>**
- To move to the newly created branch :
 - **git checkout <branch-name>**
- LPT :: To create move to the newly created branch in one command :
 - **git checkout -b <branch-name>**

```
(dcs) D:\Test\FlexSeal>git branch
* main

(dcs) D:\Test\FlexSeal>git checkout -b README
Switched to a new branch 'README'

(dcs) D:\Test\FlexSeal>
```

NOTE : If you're using VSCode, you can select and create and move among branches using a GUI in the bottom left corner that states the working branch at all times.

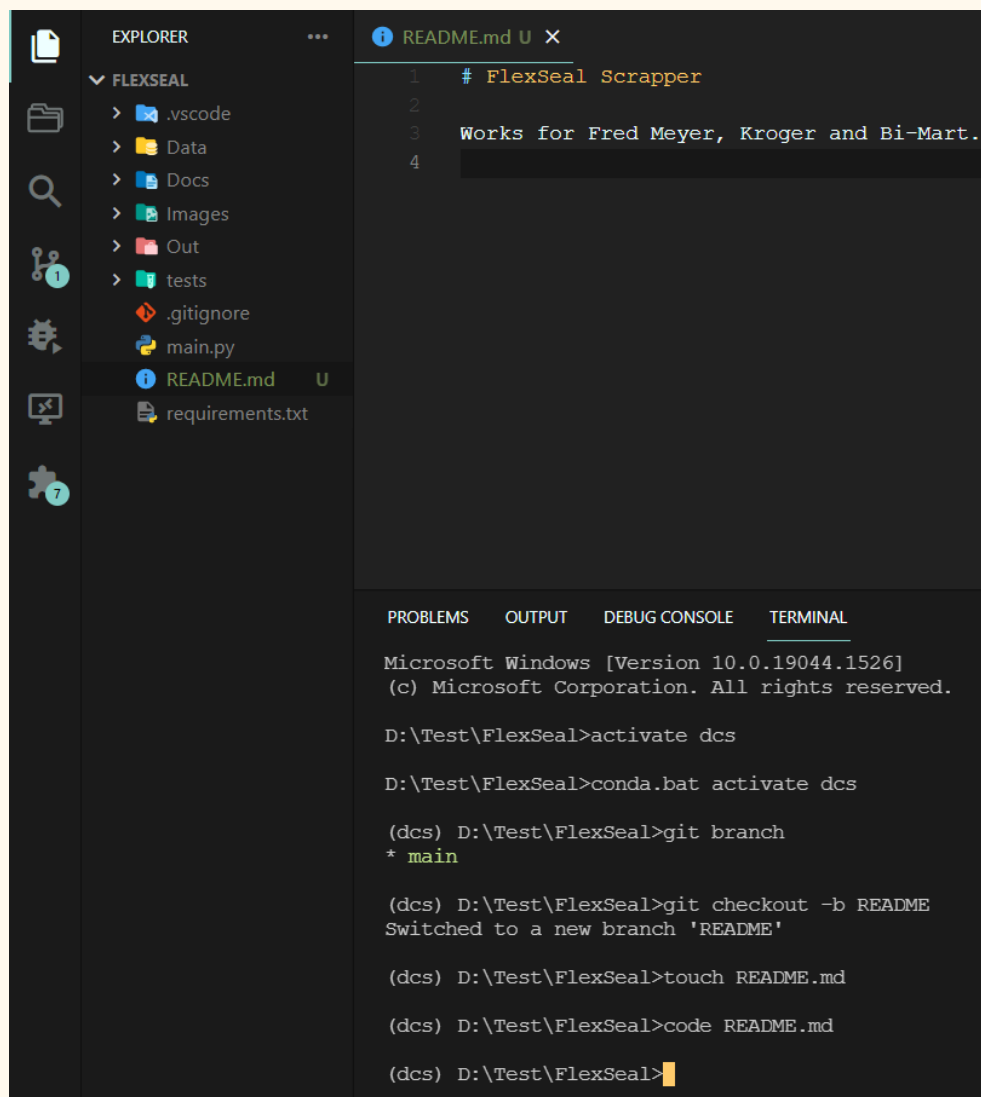
Create your files. And save your changes. Here, I have created a new README file. If you see a 'U' beside your file, that means "Updated", i.e, the file is newly created and has not been added to the repository yet.

- To add your changes:

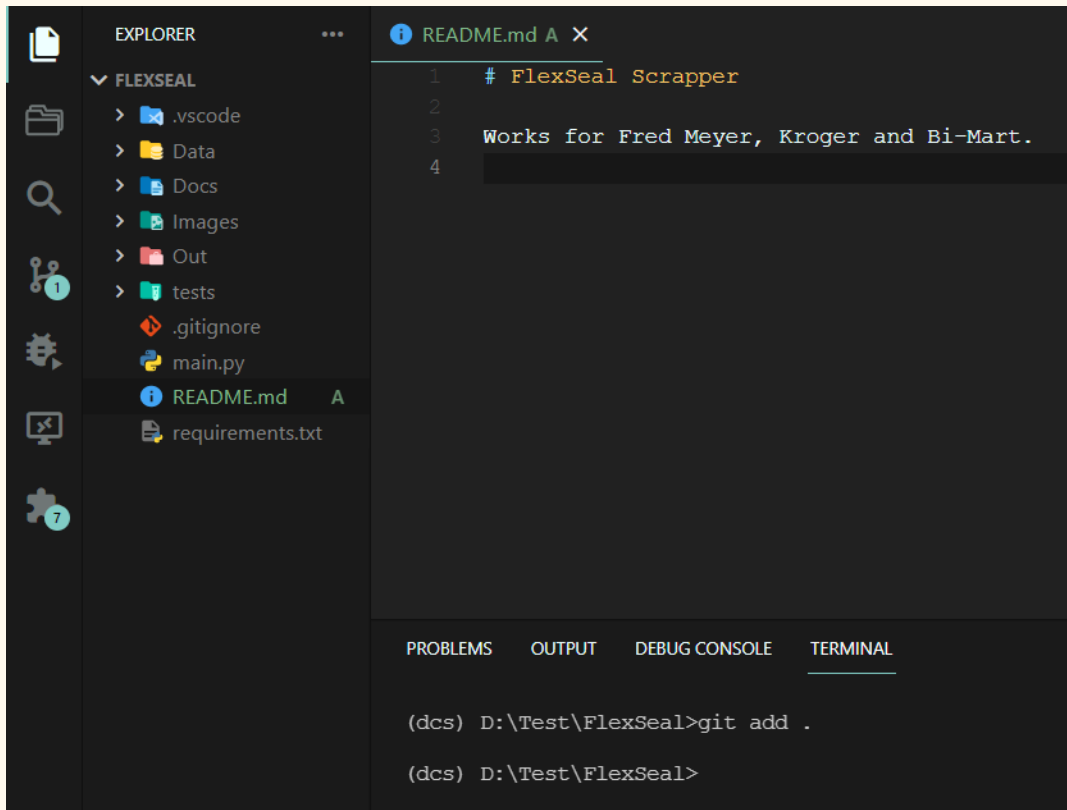
- `git add <path-to-files>`

- To add all files at once :

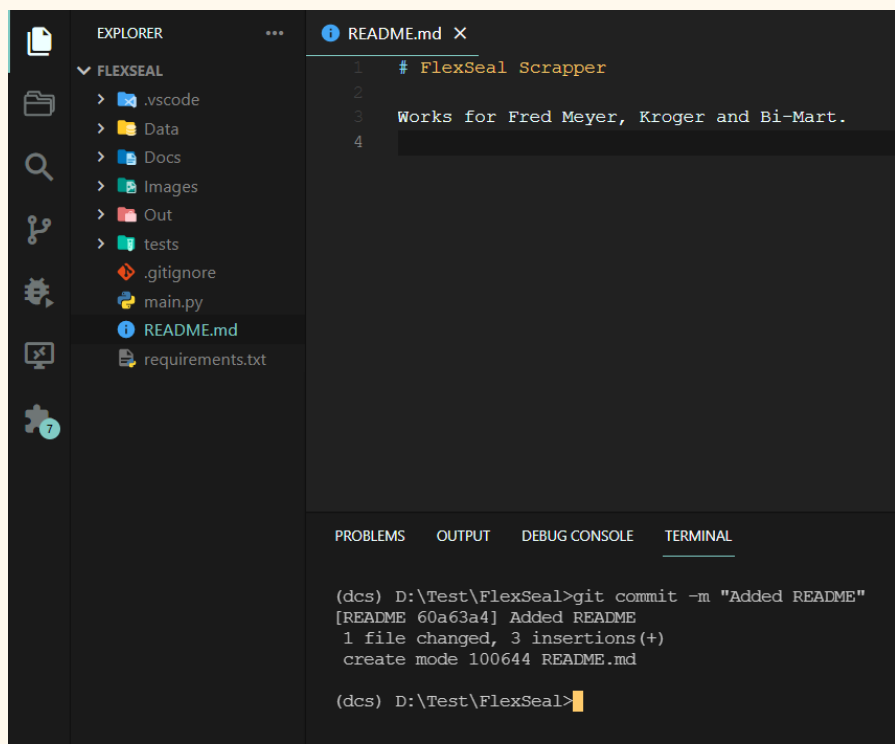
- `git add .`



The file has now been added to your repo. The 'U' has been changed to 'A', that means the file is now ADDED. You may proceed to save your changes by doing a commit.



- To commit changes
 - (Commit message is necessary while Description is optional):
 - **`git commit -m "Commit Message" "Description"`**
- LPT :: To add and commit everything in one command
 - (only works if no new files have been created):
 - **`git commit -am "Commit Message" "Description"`**

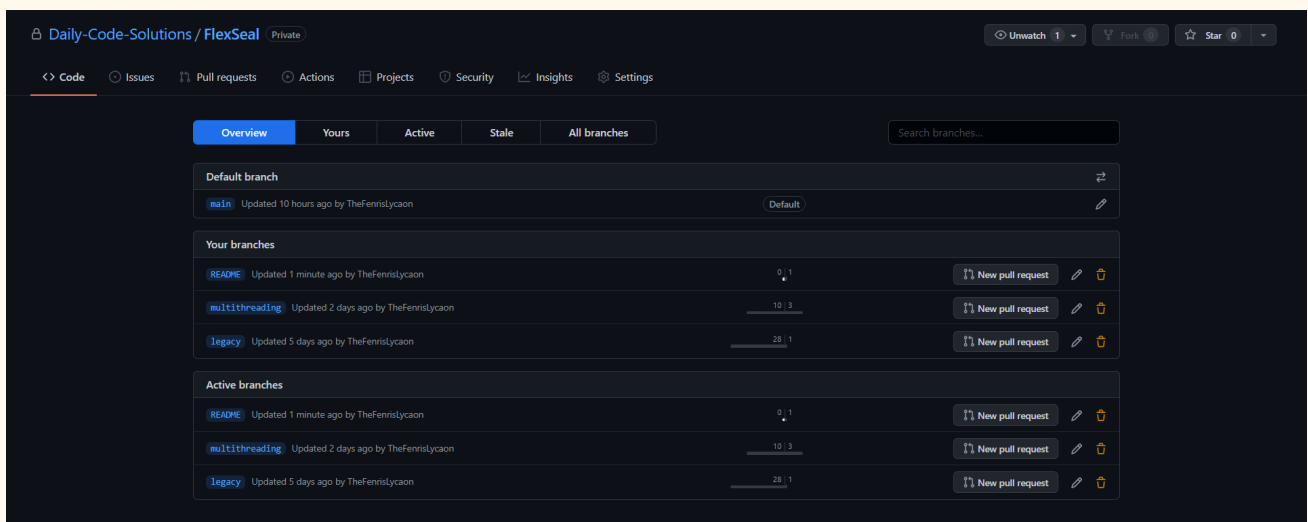


- To push your changes on remote :
 - **git push**
- If this is your first push, you need to add an upstream URL to the repo.
 - **git push --set-upstream origin <name-of-branch>**

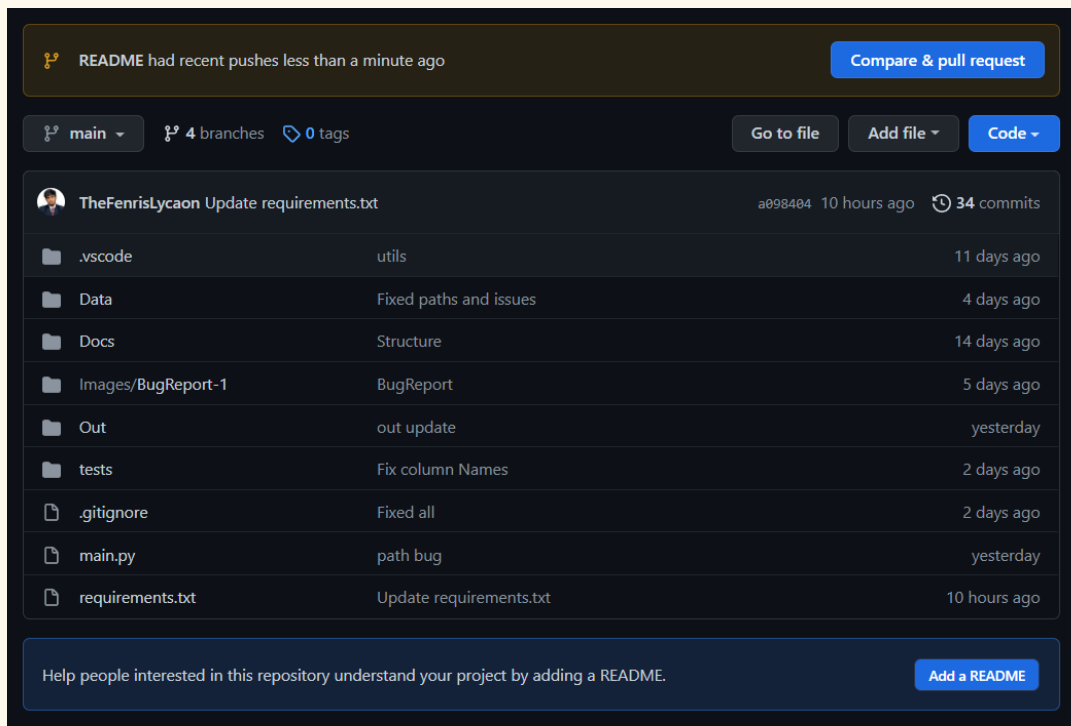
```
(dcs) D:\Test\FlexSeal>git push --set-upstream origin README
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 339 bytes | 339.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'README' on GitHub by visiting:
remote:   https://github.com/Daily-Code-Solutions/FlexSeal/pull/new/README
remote:
To https://github.com/Daily-Code-Solutions/FlexSeal.git
 * [new branch]      README -> README
branch 'README' set up to track 'origin/README'.

(dcs) D:\Test\FlexSeal>
```

Voila ! You just made your first commit ! if you head over to GitHub, you can see the branch you just created. Now you need to create a pull request. You can do this via CLI as well but I recommend using GitHub as you need to review and merge as well.



In the code section, you should see the following button.



README had recent pushes less than a minute ago [Compare & pull request](#)

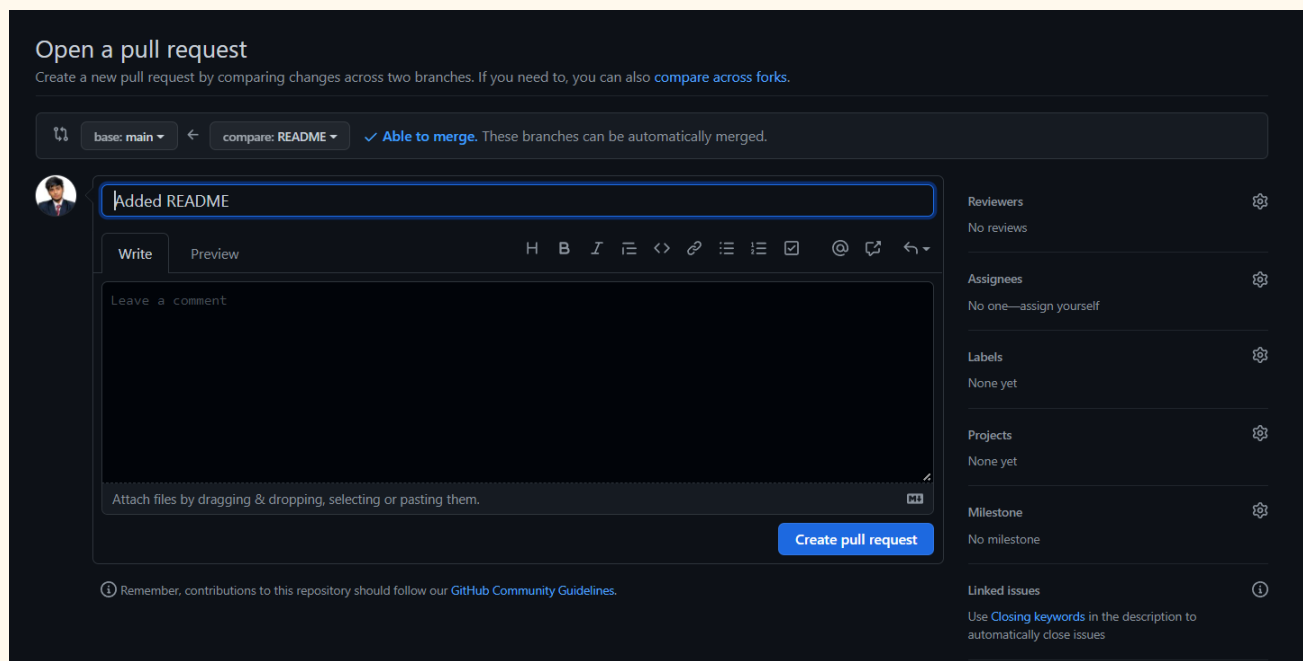
main 4 branches 0 tags [Go to file](#) [Add file](#) [Code](#)

TheFenrisLyaon Update requirements.txt a098404 10 hours ago 34 commits

| File/Folder | Commit Message | Commit Time |
|--------------------|-------------------------|--------------|
| .vscode | utils | 11 days ago |
| Data | Fixed paths and issues | 4 days ago |
| Docs | Structure | 14 days ago |
| Images/BugReport-1 | BugReport | 5 days ago |
| Out | out update | yesterday |
| tests | Fix column Names | 2 days ago |
| .gitignore | Fixed all | 2 days ago |
| main.py | path bug | yesterday |
| requirements.txt | Update requirements.txt | 10 hours ago |

Help people interested in this repository understand your project by adding a README. [Add a README](#)

Create a pull request.



Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

base: main ← compare: README ✓ [Able to merge](#). These branches can be automatically merged.

Added README

Write Preview H B I E < > Link List Checkmark @ Share Undo

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

[Create pull request](#)

Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

Reviewers No reviews

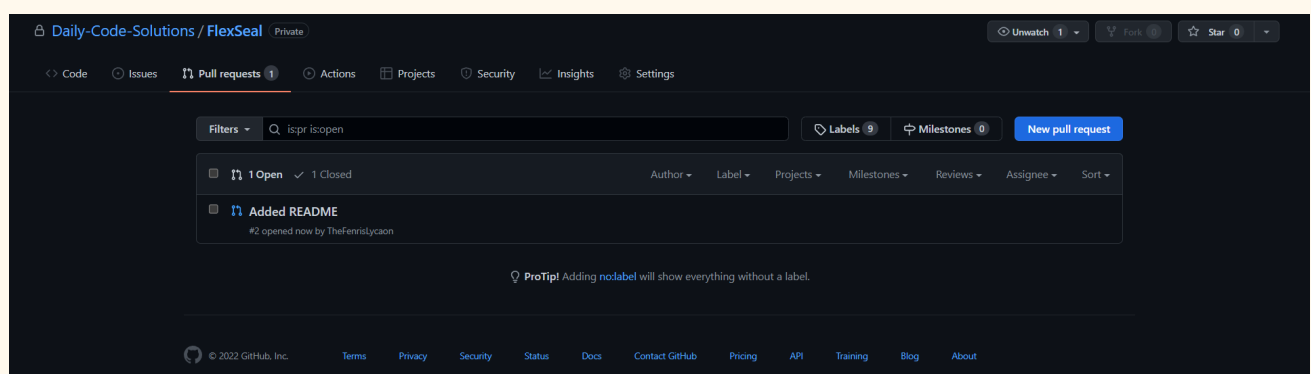
Assignees No one—assign yourself

Labels None yet

Projects None yet

Milestone No milestone

Linked issues Use [Closing keywords](#) in the description to automatically close issues



Daily-Code-Solutions / FlexSeal Private Unwatch 1 Fork 0 Star 0

Code Issues **Pull requests** 1 Actions Projects Security Insights Settings

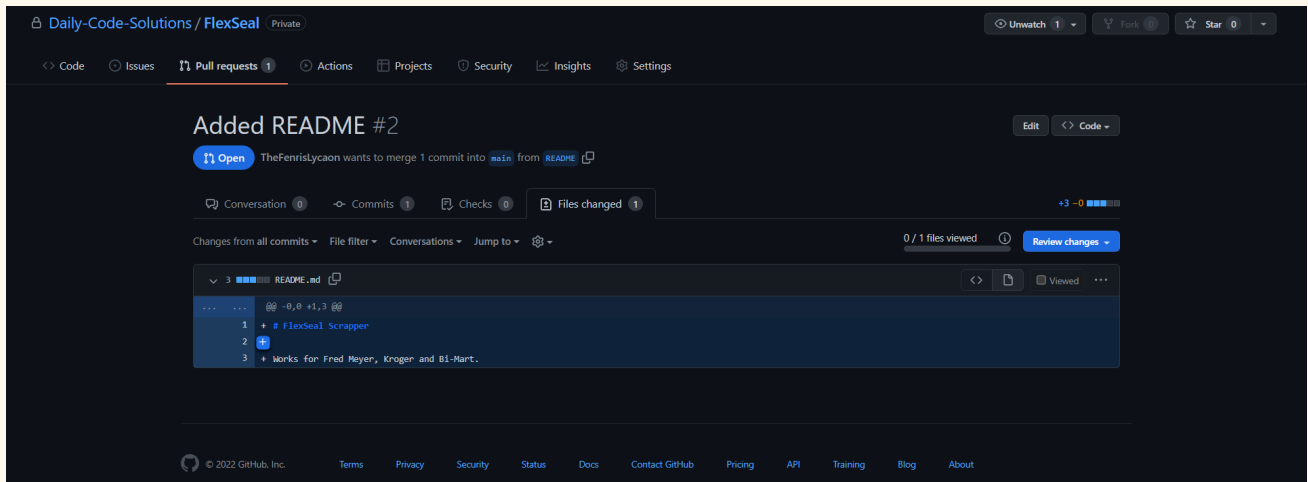
Filters ispr isopen Labels 9 Milestones 0 [New pull request](#)

| Open | 1 Open | 1 Closed | Author | Label | Projects | Milestones | Reviews | Assignee | Sort |
|--------------|---------------------------------|----------|--------|-------|----------|------------|---------|----------|------|
| Added README | #2 opened now by TheFenrisLyaon | | | | | | | | |

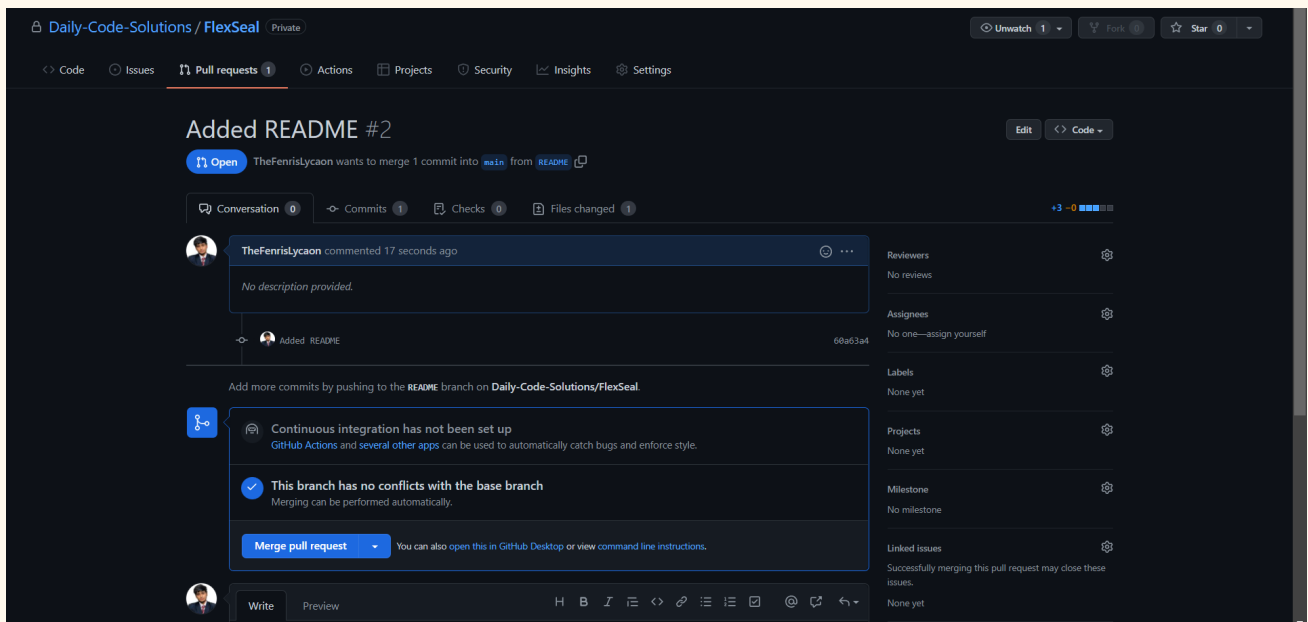
ProTip! Adding nolabel will show everything without a label.

© 2022 GitHub, Inc. Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About

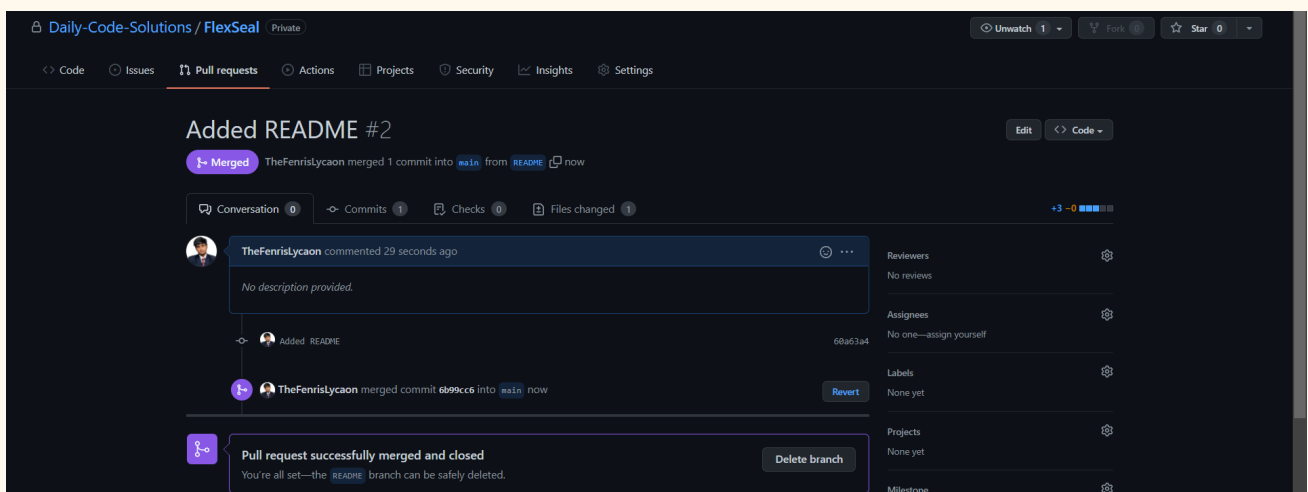
Review your code :

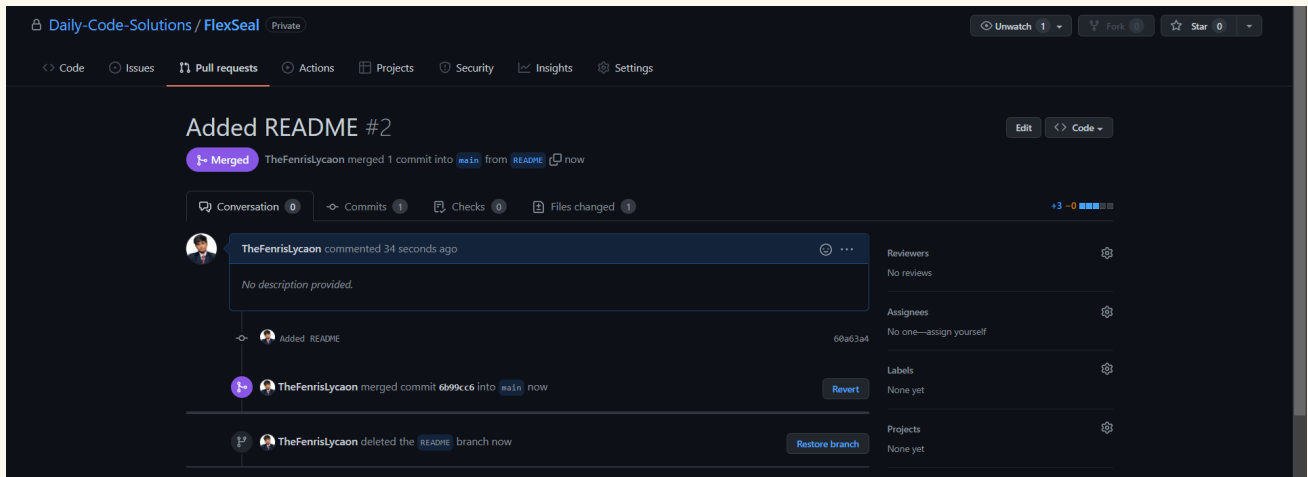


Merge the pull request :

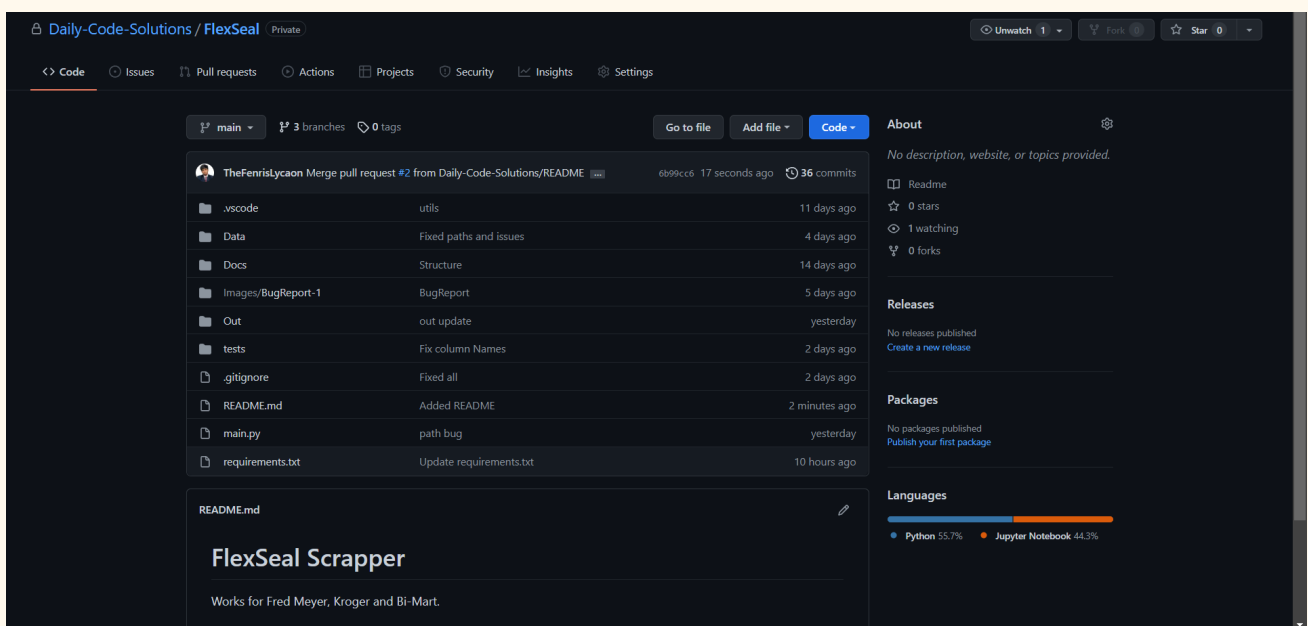


It is recommended to delete the branch if you don't have to improve this feature anymore and don't want this code separately.

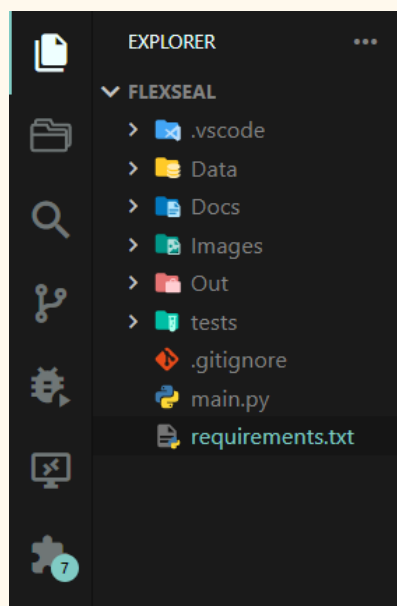




As you can see the file has been added to the main branch on the remote repository.



It is still not showing up on the local branch though. This is because you need to pull these changes.



- To sync your local repo with the remote :

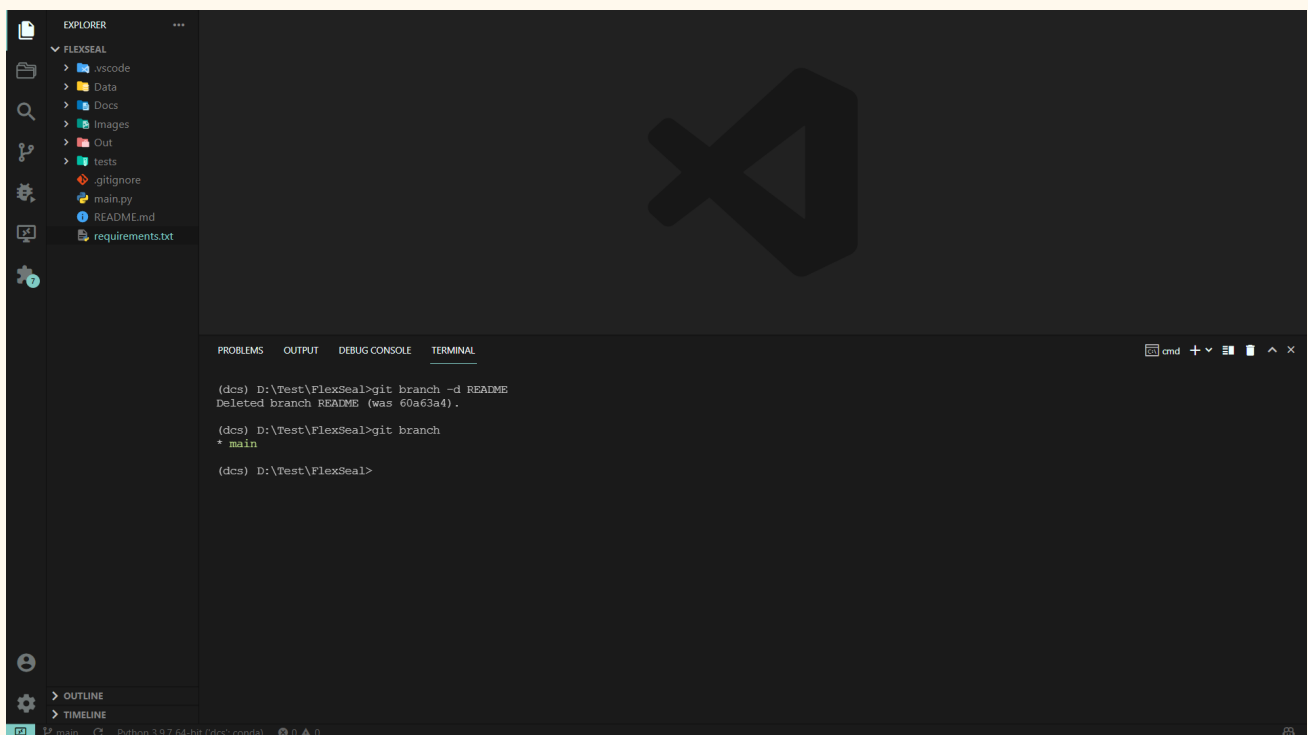
- **git pull**

```
(dcs) D:\Test\FlexSeal>git checkout main
Already on 'main'
Your branch is behind 'origin/main' by 2 commits, and can be fast-forwarded.
(use "git pull" to update your local branch)

(dcs) D:\Test\FlexSeal>git pull
Updating a098404..6b99cc6
Fast-forward
 README.md | 3 +++
 1 file changed, 3 insertions(+)
 create mode 100644 README.md

(dcs) D:\Test\FlexSeal>
```

Now you have the updated codebase. And can repeat the whole process.



- If you already have your code that you want to add to github, create a new repo on github and run these in terminal:

- **git init**
 - **git remote add origin <url>**
 - **git branch -m main**
 - **git push -u origin main**

Apart from the CLI, if you're using Visual Studio Code, the inbuilt git helper works brilliantly. Just sign in to VSC and you're ready to go.