

# The Complete Guide Git and GitHub

## Course Overview:

- 1. Introduction to Git and GitHub**
- 2. Optional Mac Terminal & Windows Command prompt Introduction**
- 3. Version Management with Git – The Basics**
- 4. Diving Deeper into the Git**
- 5. From Local to Remote Understanding Git Hub**
- 6. Git Hub Deep Drive Collaboration & Contribution**
- 7. Real Project Example Git & GitHub Applied**

## Introduction to Git and GitHub:

### What is Git?

Official site: <https://git-scm.com/>

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Control & tracking of code changes over time

### What is Version Management / Control?

Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.



Fig. Version Management / Control



Fig. Git

Git is a local tool. It's installed on your machine and therefore, only you can use it as the user of this computer. Besides that, the code you have managed in Git can only be accessed from that computer and not from anywhere else. This means if you're not on your computer, and you want to work on a project but you are somewhere else in the world, then you have a bit of a problem. Besides the fact that if you lose your computer or if it crashes that the code is lost. But that's another topic. So, Git is maybe just one part of the solution we need to manage our development projects as efficient as possible. This is where GitHub comes into play

## What is GitHub?

Official Site: <https://github.com/>

GitHub is obviously used where the world builds software. We see that millions of developers and companies build, ship, and maintain their software on GitHub. So, it's the largest and most advanced development platform in the world. So, GitHub seems to have a high focus on two developers, but what does this mean in detail now? We learnt that GitHub is the largest development platform and that many developers are using it for their projects. And why are developers doing this? Well, because GitHub is a cloud hosting and collaboration provider specifically made for developers, therefore also for web developers. It's Cloud Hosting Provider This means it's a service, which allows you to store your data, not in the local environment, but in the Cloud. It is for free for basic use cases and also the used cases we have here throughout this course. There are paid options mainly required for companies and really bigger companies to well, be able to efficiently manage the project.

GitHub is also for free. And it also is a collaboration provider. This means which Git is a local tool, GitHub with the code being available in the Cloud, also allows us to collaborate on our projects with other people. And this is very important because if you think about big companies like Facebook, for example, or also smaller companies for two, three, four, five or 10 developers work together on the same project, well, these developers or we as developers need access to this code to work on their specific parts of a website, for example. And this is what GitHub does in the end. **GitHub is a Git repository hosting provider** to be even more precise here. A Git repository is basically a Git-managed project in simple terms. So with that capabilities, so these local capabilities of Git with managing that code efficiently, with managing the history and also with tracking changes, and the capabilities of GitHub with being able to host the code on a Cloud service and to enable that great collaboration capabilities, well, this is how Git and GitHub are connected. It's very important to understand though, that GitHub and Git are not generally related. This relation only evolves because well, GitHub is the perfect addition to Git, and turns out that many people using it are also using GitHub, therefore this is why Git and GitHub sometimes are assumed to be well kind of one thing. But these are two different tools, two different services, perfectly working and perfectly used together by millions of developers in the world. So, this is what Git and GitHub are in a nutshell.



Fig. Git & GitHub.

## Optional Mac Terminal & Windows Command prompt Introduction:

### The Text Based Computer Interaction (Command Line What & Why?)

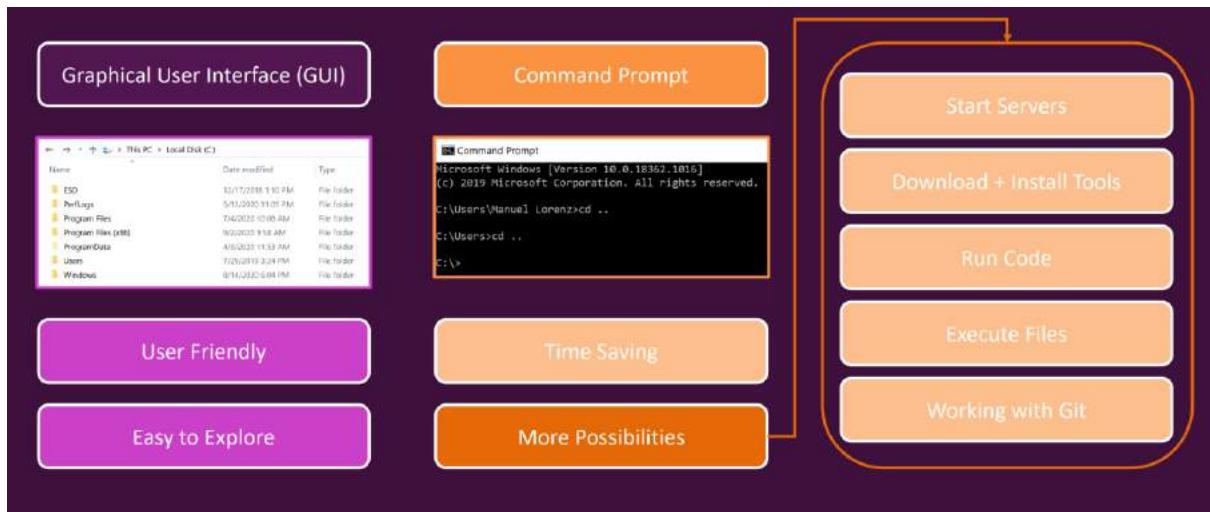


Fig. GUI vs Command Prompt

## Comparing Mac & Windows Command Line:

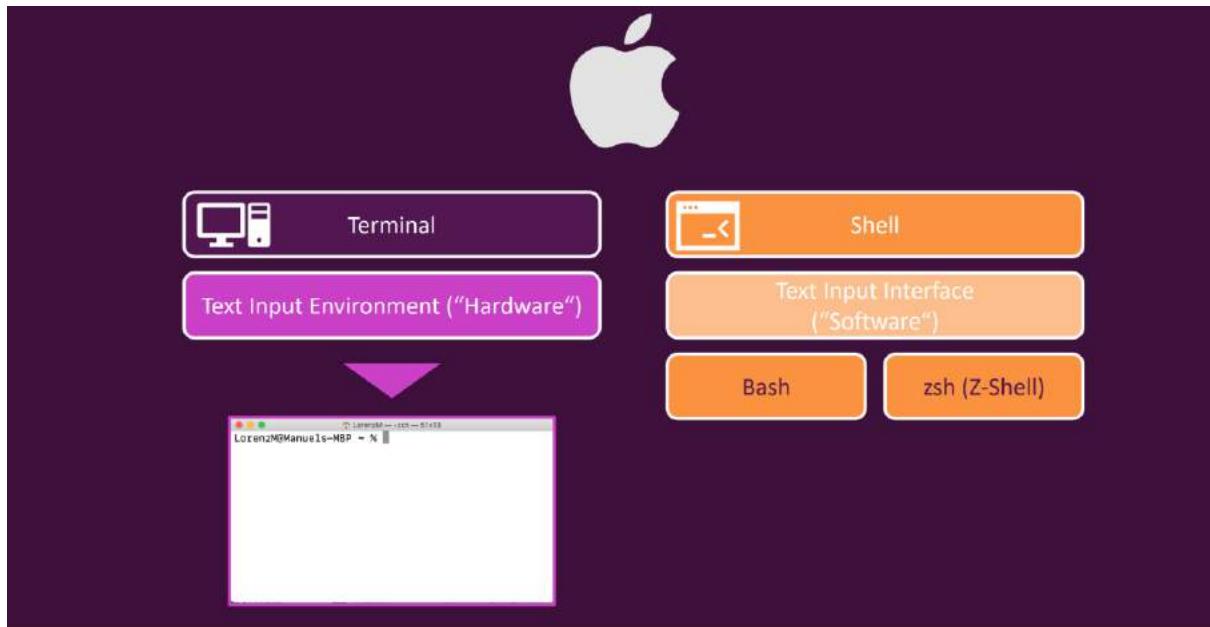


Fig. Mac Terminology

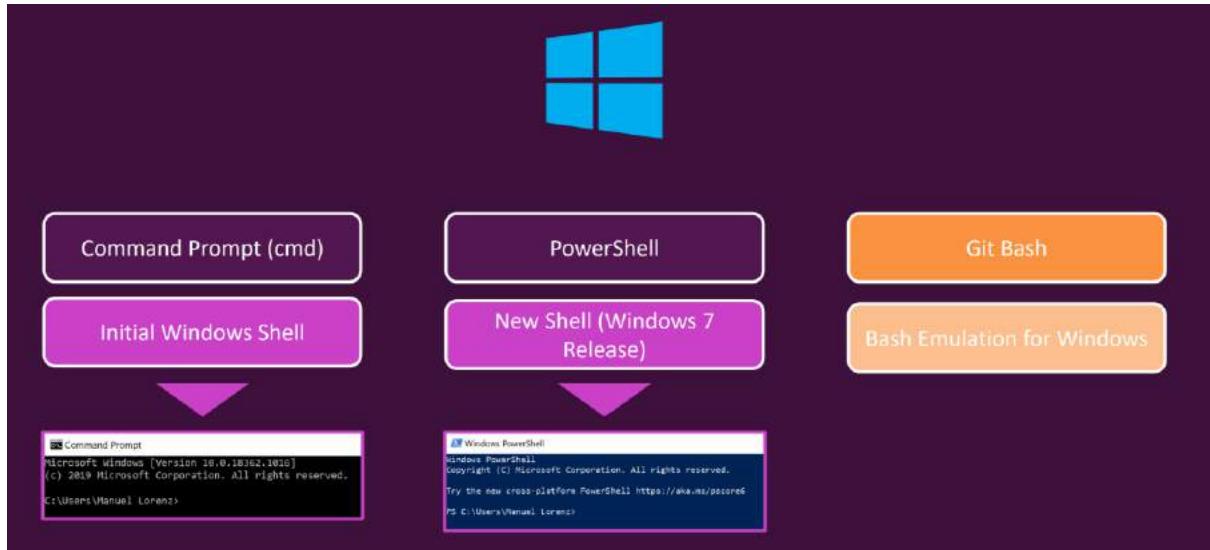


Fig. Windows Terminology



Fig. Command Line Tools

### Mac Terminal:



Fig. Mac Terminal (z-Shell Command)

Note: This Command Used in Mac-Book for Mac User.

## Windows Command Prompt:

Command Prompt is the traditional Command Line Interface

### The Basics:

- **dir: lists items**

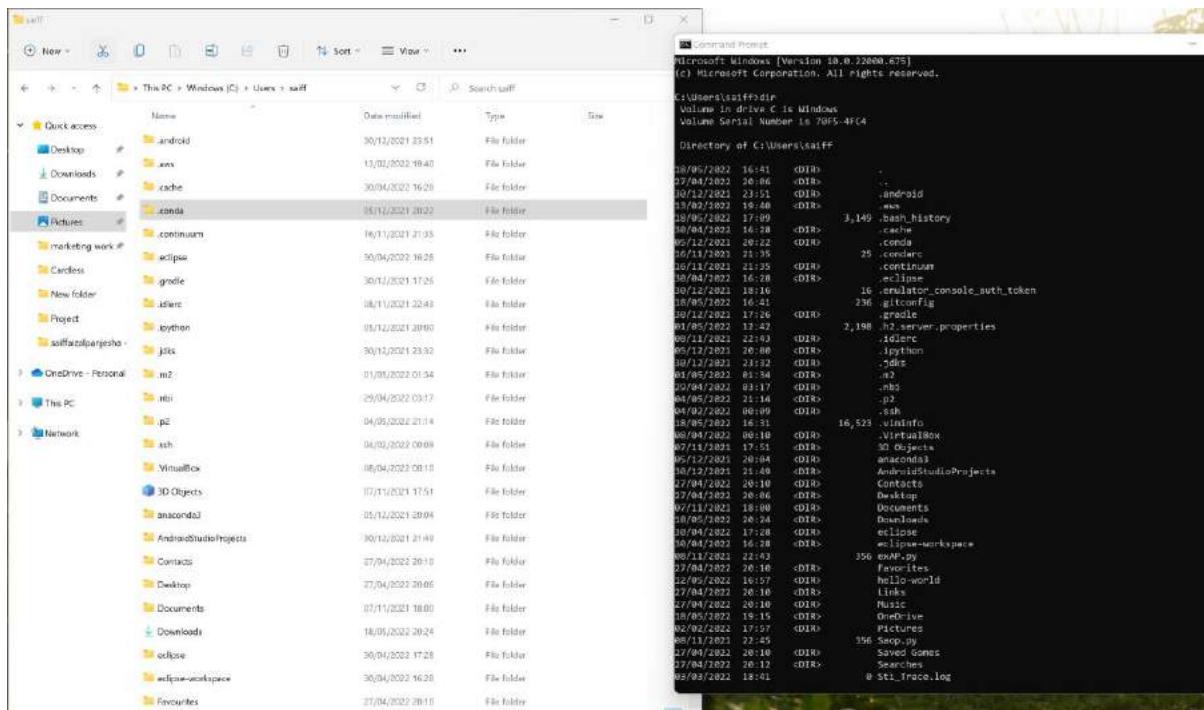


Fig. Lists of Items

- **cd(..): Change Directory**

```

C:\Users\saifff>cd ..
C:\Users>

```

Output of the command:

```

27/04/2022 23:51 <DIR> .android
18/03/2022 19:00 <DIR> .aws
18/05/2022 17:09 <DIR> .hash_history
18/04/2022 16:28 <DIR> .cache
05/12/2021 20:22 <DIR> .conda
16/11/2021 21:35 <DIR> .conjur
15/11/2021 21:35 <DIR> .continuum
18/04/2022 16:16 <DIR> .eclipse
18/04/2022 16:41 <DIR> .gitconfig_console_auth_token
10/12/2022 17:26 <DIR> .gitconfig_gredie
01/05/2022 12:42 <DIR> .h2.server.properties
05/11/2021 22:43 <DIR> .idclic
05/12/2021 20:00 <DIR> .ipython
10/12/2021 20:00 <DIR> .jaks
01/05/2022 21:34 <DIR> .m2
19/04/2022 03:17 <DIR> .nbd
04/05/2022 21:14 <DIR> .p2
04/02/2022 00:09 <DIR> .ssh
18/05/2022 16:31 <DIR> 16,523 .vininfo
05/04/2022 00:10 <DIR> .VirtualBox
07/04/2022 17:00 <DIR> 30 .objdump
05/12/2021 18:04 <DIR> .AndroidStudioProjects
18/12/2021 21:00 <DIR> .AndroidStudioProjects
27/04/2022 20:10 <DIR> .Contacts
27/04/2022 20:06 <DIR> .Desktop
07/11/2021 18:00 <DIR> .Documents
18/05/2022 20:34 <DIR> .Downloads
18/04/2022 17:00 <DIR> .eclipse
18/04/2022 16:18 <DIR> .eclipse-workspace
09/11/2021 22:43 <DIR> 356 .exp4j
27/04/2022 20:10 <DIR> .Favorites
12/05/2022 16:57 <DIR> .hello-world
27/04/2022 20:10 <DIR> .Links
27/04/2022 20:10 <DIR> .Music
18/09/2021 19:10 <DIR> .OneDrive
07/04/2022 17:17 <DIR> .Pictures
08/11/2021 22:05 <DIR> 556 .Scap.py
27/04/2022 20:10 <DIR> .Saved Games
17/04/2022 20:12 <DIR> .Searches
03/03/2022 18:41 <DIR> 0 .Stl_Trace.log
01/05/2022 12:14 <DIR> 28,672 .test-mv.db
01/05/2022 12:10 <DIR> 579 .test-trace.db
18/04/2022 19:59 <DIR> .Videos
27/04/2022 20:10 <DIR> .Videos
28/03/2022 14:47 <DIR> .VirtualBox VMs
11 File(s) 52,110 bytes
37 Dir(s) 71,163,752,448 bytes free
C:\Users\saifff>cd ..
C:\Users>

```

Fig. Change Directory to Users

```

C:\Users\saifff>cd ..
C:\Users>

```

Output of the command:

```

C:\Users>dir
Volume in drive C is Windows
Volume Serial Number is 70F5-4FC4

Directory of C:\Users

07/04/2022 20:06 <DIR> Favorites
12/04/2022 16:17 <DIR> Microsoft
17/04/2022 16:30 <DIR> Links
12/04/2022 20:10 <DIR> Music
18/05/2022 19:15 <DIR> OneDrive
02/02/2022 17:57 <DIR> Pictures
08/11/2021 22:45 <DIR> 356 .Scap.py
27/04/2022 20:10 <DIR> .Saved Games
17/04/2022 20:10 <DIR> .Searches
03/03/2022 18:41 <DIR> 0 .Stl_Trace.log
01/05/2022 12:14 <DIR> 28,672 .test-mv.db
01/05/2022 12:10 <DIR> 579 .test-trace.db
18/04/2022 19:59 <DIR> .Videos
27/04/2022 20:10 <DIR> .Videos
28/03/2022 14:47 <DIR> .VirtualBox VMs
11 File(s) 52,110 bytes
37 Dir(s) 71,163,752,448 bytes free
C:\Users>cd ..
C:\>dir
Volume in drive C is Windows
Volume Serial Number is 70F5-4FC4

Directory of C:\

01/05/2022 21:01 <DIR> apache-maven-3.8.5-bin
09/05/2022 03:40 <DIR> Intel
28/04/2022 01:34 <DIR> PerfLogs
15/05/2021 17:40 <DIR> Program Files
15/05/2021 17:40 <DIR> Program Files (x86)
12/04/2022 20:06 <DIR> User
14/05/2022 14:45 <DIR> Windows
27/04/2022 20:10 <DIR> Windows.old
11 File(s) 0 bytes
8 Dir(s) 71,159,087,104 bytes free
C:\>

```

Fig. Change Directory to C drive

## Changing to D- Drive:

C:\>D:

D:\>

- Relative Paths:

```
D:\>cd Devops
```

```
D:\Devops>dir  
Volume in drive D is Data  
Volume Serial Number is 04F5-839B
```

```
Directory of D:\Devops
```

```
18/05/2022 18:08 <DIR> .  
02/05/2022 17:35 <DIR> hello-world  
18/05/2022 18:08 <DIR> Project  
18/05/2022 17:49 <DIR> saiffaizalpanjesha -aws  
0 File(s) 0 bytes  
4 Dir(s) 275,885,756,416 bytes free
```

- Absolute Path:

```
D:\Devops>cd D:\Devops\saiffaizalpanjesha -aws
```

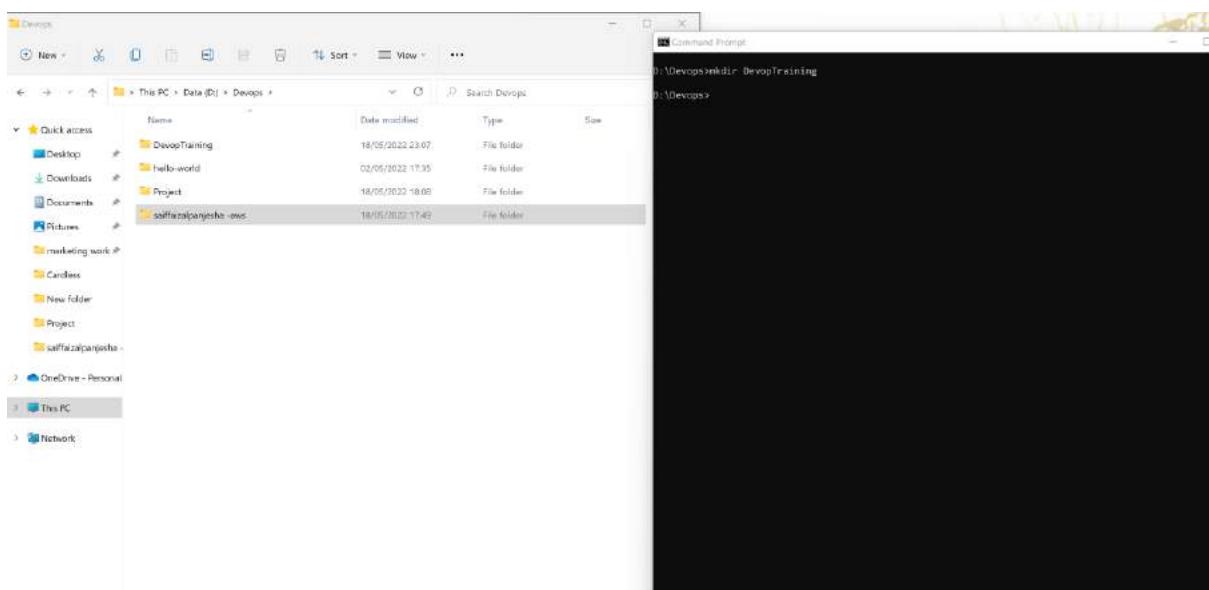
```
D:\Devops\saiffaizalpanjesha -aws>dir  
Volume in drive D is Data  
Volume Serial Number is 04F5-839B  
Directory of D:\Devops\saiffaizalpanjesha -aws  
18/05/2022 17:49 <DIR> .  
18/05/2022 18:08 <DIR> ..  
12/04/2022 08:49 26,740 3646_Skills.pdf  
18/05/2022 17:49 252 Devops Other Courses.txt  
02/05/2022 00:01 1,700 Devops_Project_Key.pem  
18/05/2022 02:25 176 devpos.txt  
01/05/2022 23:47 135 new_user_credentials.csv  
03/05/2022 21:57 1,868 Password.txt  
6 File(s) 30,871 bytes  
2 Dir(s) 275,885,756,416 bytes free
```

- **cls :(Clear command prompt)**



Fig. Clear Screen of Cmd Prompt

- **mkdir : Creates Folder**
- **echo our Devops File > text.txt : Creating a File**



```
D:\Devops\DevopTraining>echo our Devops File > text.txt

D:\Devops\DevopTraining>dir
 Volume in drive D is Data
 Volume Serial Number is 04F5-839B

 Directory of D:\Devops\DevopTraining

18/05/2022  23:11      <DIR>          .
18/05/2022  23:07      <DIR>          ..
18/05/2022  23:11                  18  text.txt
                           1 File(s)           18 bytes
                           2 Dir(s)  275,885,752,320 bytes free

D:\Devops\DevopTraining>type text.txt
our Devops File

D:\Devops\DevopTraining>
```

Fig. Create a Folder and File ( Devop Training and text)

- **del : Deleting File**

```
D:\Devops\DevopTraining>del text.txt

D:\Devops\DevopTraining>dir
Volume in drive D is Data
Volume Serial Number is 04F5-839B

Directory of D:\Devops\DevopTraining

18/05/2022  23:15      <DIR>          .
18/05/2022  23:07      <DIR>          ..
              0 File(s)           0 bytes
              2 Dir(s)  275,885,539,328 bytes free

D:\Devops\DevopTraining>
```

Fig. Deleted File Text

- **rmdir: Deleting Folder**

```
D:\Devops\DevopTraining>cd ..

D:\Devops>rmdir DevopTraining

D:\Devops>dir
Volume in drive D is Data
Volume Serial Number is 04F5-839B

Directory of D:\Devops

18/05/2022  23:18      <DIR>          .
02/05/2022  17:35      <DIR>          hello-world
18/05/2022  18:08      <DIR>          Project
18/05/2022  17:49      <DIR>          saiffaizalpanjesha -aws
                  0 File(s)           0 bytes
                  4 Dir(s)  275,885,506,560 bytes free

D:\Devops>
```

Fig. Deleted Folder Devop Training

- **copy : Copying Files**

```
D:\Devops>mkdir others
D:\Devops>mkdir main
D:\Devops>echo Hello Miss > test2.txt
D:\Devops>dir
Volume in drive D is Data
Volume Serial Number is 04F5-8398

Directory of D:\Devops

18/05/2022 23:23    <DIR>        .
02/05/2022 17:35    <DIR>        hello-world
18/05/2022 23:23    <DIR>        main
18/05/2022 23:23    <DIR>        others
18/05/2022 18:08    <DIR>        Project
18/05/2022 17:49    <DIR>        saiffaizalpanjehsa -aws
18/05/2022 23:23          13 test2.txt
                           1 File(s)           13 bytes
                           6 Dir(s) 275,885,506,560 bytes free

D:\Devops>type test2
The system cannot find the file specified.

D:\Devops>type test2.txt
Hello Miss

D:\Devops>copy test2.txt main
      1 file(s) copied.

D:\Devops>
```

Fig. Copying Files

- **move: Move the Files**

```
D:\Devops\main>del test2.txt
D:\Devops\main>dir
Volume in drive D is Data
Volume Serial Number is 04F5-8398

Directory of D:\Devops\main

18/05/2022 23:30    <DIR>        .
18/05/2022 23:23    <DIR>        ..
                           0 File(s)           0 bytes
                           2 Dir(s) 275,885,371,392 bytes free

D:\Devops\main>cd ..
D:\Devops>move test2.txt main
      1 file(s) moved.

D:\Devops>dir
Volume in drive D is Data
Volume Serial Number is 04F5-8398

Directory of D:\Devops

18/05/2022 23:31    <DIR>        .
02/05/2022 17:35    <DIR>        hello-world
18/05/2022 23:31    <DIR>        main
18/05/2022 23:23    <DIR>        others
18/05/2022 18:08    <DIR>        Project
18/05/2022 17:49    <DIR>        saiffaizalpanjehsa -aws
                           0 File(s)           0 bytes
                           6 Dir(s) 275,885,371,392 bytes free

D:\Devops>cd main

D:\Devops\main>dir
Volume in drive D is Data
Volume Serial Number is 04F5-8398

Directory of D:\Devops\main

18/05/2022 23:31    <DIR>        .
18/05/2022 23:31    <DIR>        ..
18/05/2022 23:23          13 test2.txt
                           1 File(s)           13 bytes
                           2 Dir(s) 275,885,371,392 bytes free

D:\Devops\main>type test2.txt
Hello Miss
```

Fig. Moving Files

# Version Management with Git – The Basics

## Contents:

- **Theory – How Git Works?**
- **Installation & Deployment Environments**
- **Repositories, Branches and Commit**

## How Git Works?

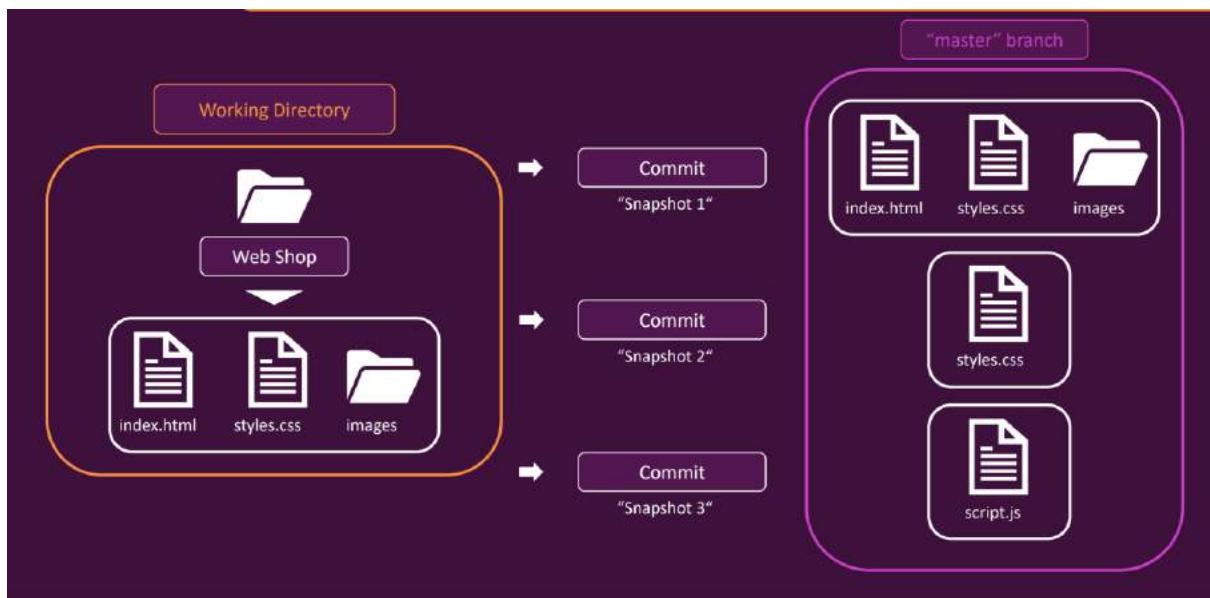


Fig. Working of Git.

## Working Directory vs Repository

### Git under the Hood:



Fig. Git Under the Hood (Directory & Repositories)

## Understanding Branches & Commit



Fig. Branches & Commit

## **Installing Git on Windows:**

Go to official website: <https://git-scm.com/>

1. Steps For Installing Git for Windows. Download Git for Windows. Extract and Launch Git Installer. Server Certificates, Line Endings and Terminal Emulators. ...
2. How to Launch Git in Windows. Launch Git Bash Shell. Launch Git GUI.
3. Connecting to a Remote Repository. Create a Test Directory. Configure GitHub Credentials.



```
git Select Command Prompt
Microsoft Windows [Version 10.0.22000.675]
(c) Microsoft Corporation. All rights reserved.

C:\Users\saiff>git --version
git version 2.35.1.windows.2

C:\Users\saiff>
```

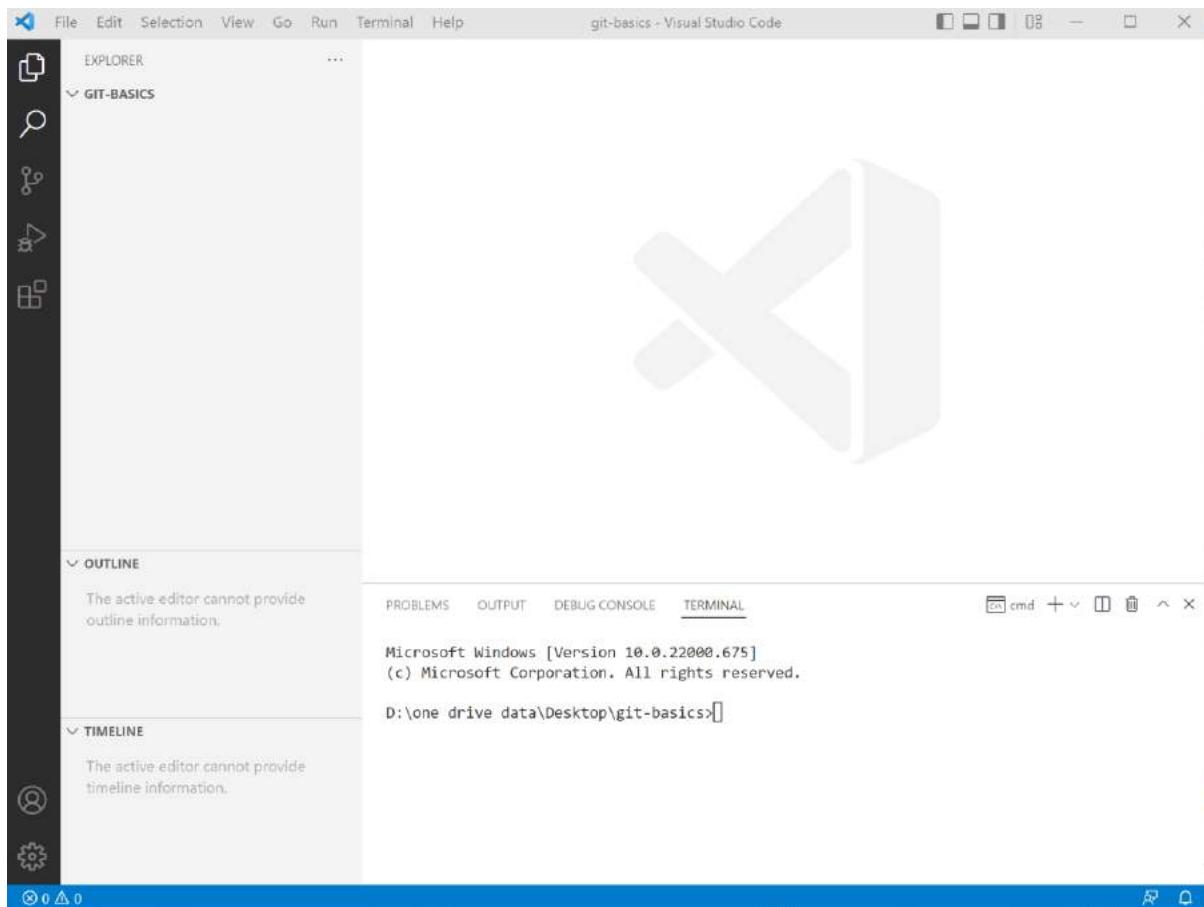
Fig. Successful Installed Git version 2.35.1

## **Installing Visual Studio Code:**

Go to Website: <https://code.visualstudio.com/>

Step 1: Download VS code from here Link.

Step 2: Download the Visual Studio Code installer for Windows. Once it is downloaded, run the installer (VSCodeUserSetup-{version}.exe). Then, run the file – it will only take a minute. Accept the agreement and click “next.”



## Fig. Visual Studio Install Success

## **Initializing the repository & creating the First Commit (“git init” and “git commit”):**

D:\one drive data\Desktop\git-basics>git --version

git version 2.35.1.windows.2

D:\one drive data\Desktop\git-basics>git status

**fatal: not a git repository (or any of the parent directories): .git**

```
D:\one drive data\Desktop\git-basics>
```

**git init : Initialize the Git**

```
D:\one drive data\Desktop\git-basics>git init
```

```
Initialized empty Git repository in D:/one drive data/Desktop/git-basics/.git/
```

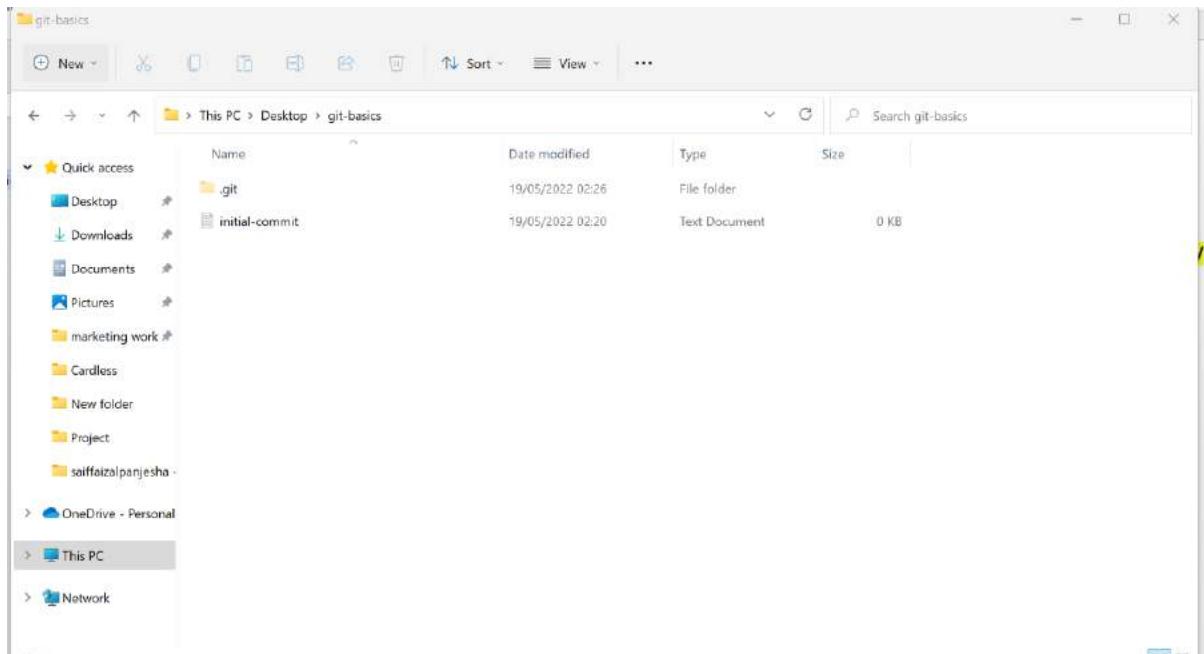


Fig. git hidden folder created after initialized

**Untracked Files :**

```
D:\one drive data\Desktop\git-basics>git status
```

```
On branch master
```

```
No commits yet
```

```
Untracked files:
```

(use "git add <file>..." to include in what will be committed)

initial-commit.txt

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

### Tracking the File:

D:\one drive data\Desktop\git-basics>git add . //Adding the files

D:\one drive data\Desktop\git-basics>git status

On branch master

No commits yet

### Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: initial-commit.txt // initiated after added

### Configure your Git username/email

1. Open the command line.
2. Set your username: git config --global user.name "FIRST\_NAME LAST\_NAME"
3. Set your email address: git config --global user.email "MY\_NAME@example.com"

D:\one drive data\Desktop\git-basics>git commit -m "updated file.txt"

[master (root-commit) 3207266] updated file.txt

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 initial-commit.txt

```
D:\one drive data\Desktop\git-basics>git status
```

```
On branch master
```

```
nothing to commit, working tree clean
```

## Diving Deeper into Commit with “git log”

To Check where is Commit History?

```
D:\one drive data\Desktop\git-basics>git log
```

```
commit 320726669068faa962f8e245df7e3be52c76accc (HEAD -> master)
```

```
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
```

```
Date: Thu May 19 02:36:24 2022 +0530
```

```
updated file.txt
```

Create a second Commit File:



Fig. Second Commit in git

The screenshot shows the Visual Studio Code interface. On the left, there's an 'OUTLINE' view which says 'No symbols found in document 'second-commit.txt''. Below it is a 'TIMELINE' view for 'second-commit.txt' stating 'No timeline information was provided'. The main area is the 'TERMINAL' tab, which displays the following command-line session:

```
Microsoft Windows [Version 10.0.22000.675]
(c) Microsoft Corporation. All rights reserved.

D:\one drive data\Desktop\git-basics>git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    second-commit.txt

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

D:\one drive data\Desktop\git-basics>
```

Fig. git status for untracked file

The screenshot shows the Visual Studio Code interface. The 'EXPLORER' sidebar shows a folder named 'GIT-BASICS' containing 'initial-commit.txt' and 'second-commit.txt'. The 'TERMINAL' tab displays the following command-line session:

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

D:\one drive data\Desktop\git-basics>git add second-commit.txt

D:\one drive data\Desktop\git-basics>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   second-commit.txt

D:\one drive data\Desktop\git-basics>
```

Fig. "git add" to track file

**The git commit command captures a snapshot of the project's currently staged changes.**

The screenshot shows the Visual Studio Code interface. The 'TERMINAL' tab displays the following command-line session:

```
D:\one drive data\Desktop\git-basics>git commit -m "added second-commit.txt"
[master dc4a671] added second-commit.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 second-commit.txt

D:\one drive data\Desktop\git-basics>
```

Fig. Staged Changes “git -commit”

```

create mode 100644 second-commit.txt

D:\one drive data\Desktop\git-basics>git log
commit dc4a67141cb3c3529cdc7da6200e9600f0c56387 (HEAD -> master)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Thu May 19 15:02:23 2022 +0530

    added second-commit.txt

commit 320726669068faa962f8e245df7e3be52c76accc
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Thu May 19 02:36:24 2022 +0530

    updated file.txt

```

D:\one drive data\Desktop\git-basics>

Fig. “git log “jump back to history”

## To check Back the previous commit

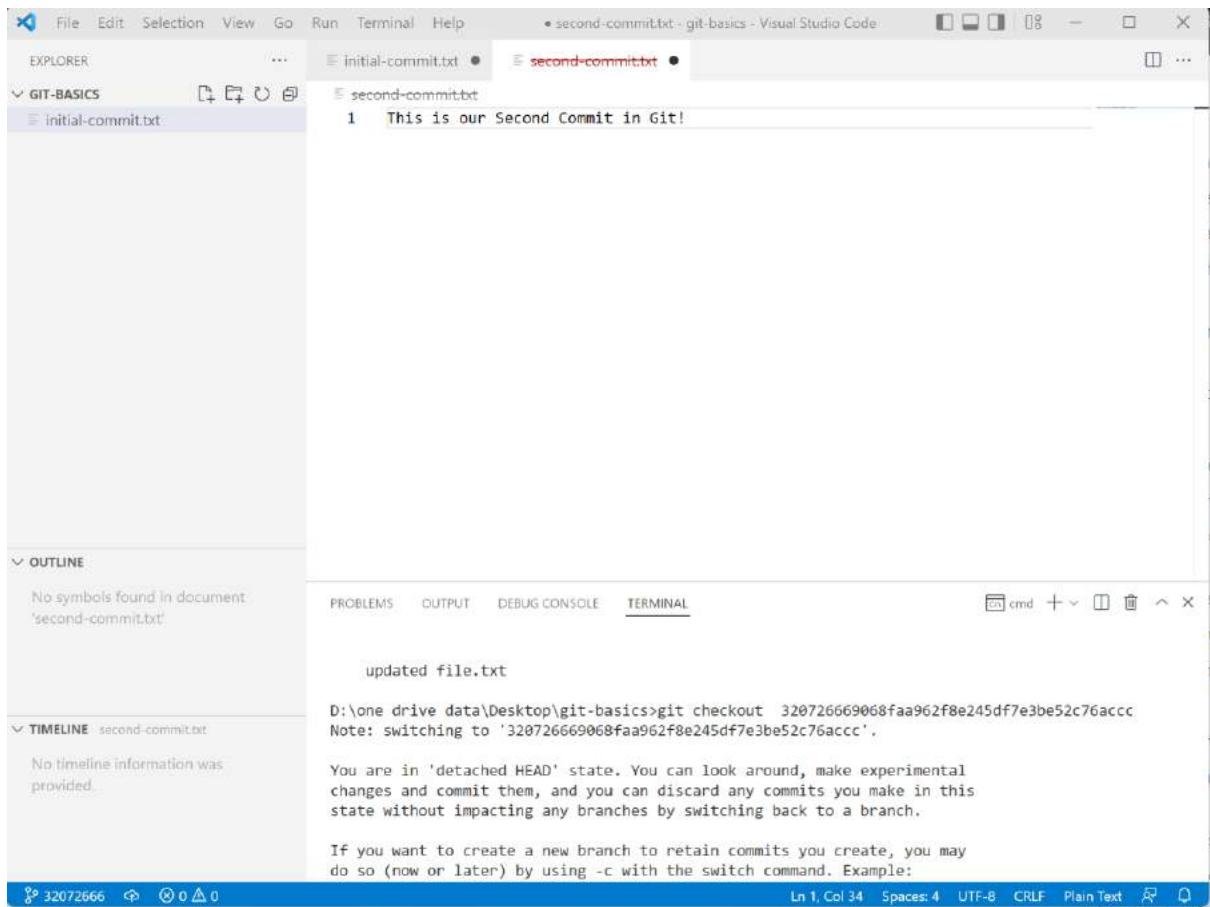


Fig. “Checkout our initial commit”

```
HEAD is now at 3207266 updated file.txt
D:\one drive data\Desktop\git-basics>git log
commit 320726669068faa962f8e245df7e3be52c76accc (HEAD)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date: Thu May 19 02:36:24 2022 +0530

    updated file.txt
```

Fig. Not updated the second commit in history

**Can the Second Commit is deleted? As we back to previous commit?**

**The Answer is No not deleted.**

**To check go to master branch and again check the history.**

```
D:\one drive data\Desktop\git-basics>git checkout master
Previous HEAD position was 3207266 updated file.txt
Switched to branch 'master'

D:\one drive data\Desktop\git-basics>git log
commit dc4a67141cb3c3529cdc7da6200e9600f0c56387 (HEAD -> master)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date: Thu May 19 15:02:23 2022 +0530

    added second-commit.txt

commit 320726669068faa962f8e245df7e3be52c76accc
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date: Thu May 19 02:36:24 2022 +0530

    updated file.txt

D:\one drive data\Desktop\git-basics>
```

Fig. Second commit is visible.

## Understanding and Creating Branches

**Branch:** is a unique set of specific code changes

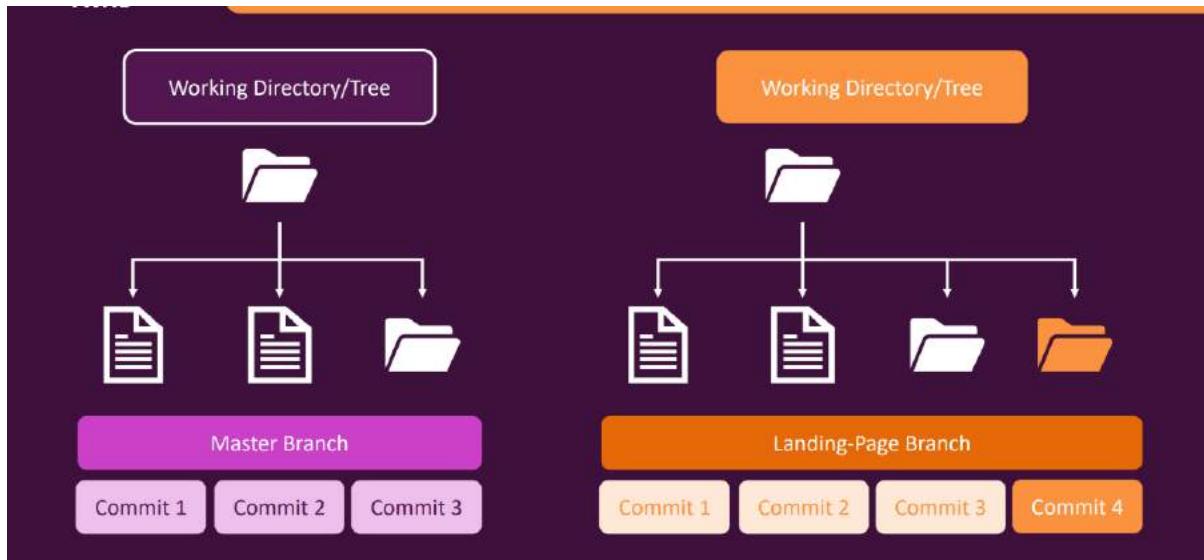


Fig. Goal to create Such copy of master branches

To check of all branches our current project:

A screenshot of Visual Studio Code showing the terminal output of the `git branch` command. The terminal window displays the following text:

```
D:\One Drive Data\Desktop\git-basics>git branch
* master
```

The left sidebar shows a file tree with 'initial-commit.txt' and 'second-commit.txt' under a 'GIT-BASICS' folder. The bottom left corner shows a timeline entry: 'added second-commit.txt... 15 mins'.

Fig. Default Branch in current project-\* master

## Creating Second Branch Landing-Page Branch

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left contains two files: 'initial-commit.txt' and 'second-commit.txt'. The Terminal tab at the top is active, displaying the command-line session:

```
D:\one drive data\Desktop\git-basics>git branch
* master

D:\one drive data\Desktop\git-basics>git branch Landing-Page

D:\one drive data\Desktop\git-basics>git branch
  Landing-Page
* master

D:\one drive data\Desktop\git-basics>
```

Fig. Second Branch is Created

## For accessing Second Branch git checkout branch-name:

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left contains two files: 'initial-commit.txt' and 'second-commit.txt'. The Terminal tab at the top is active, displaying the command-line session:

```
D:\one drive data\Desktop\git-basics>git branch
  Landing-Page
* master

D:\one drive data\Desktop\git-basics>git checkout Landing-Page
Switched to branch 'Landing-Page'

D:\one drive data\Desktop\git-basics>git branch
* Landing-Page
  master

D:\one drive data\Desktop\git-basics>
```

Fig. Accessing Second Branch in git

## Shortcut for this type of Operation:

The screenshot shows a terminal window with the following command-line session:

```
D:\one drive data\Desktop\git-basics>git checkout -b quickfix
Switched to a new branch 'quickfix'

D:\one drive data\Desktop\git-basics>git log
commit dc4a67141cb3c3529cdc7da6200e9600f0c56387 (HEAD -> quickfix, master, Landing-Page)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Thu May 19 15:02:23 2022 +0530

    added second-commit.txt

commit 320726669068faa962f8e245df7e3be52c76accc
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Thu May 19 02:36:24 2022 +0530

    updated file.txt
```

Fig. Created third branch

## Create new File Working with Branches



A screenshot of the Visual Studio Code interface. The title bar shows 'Working\_with\_Branches.txt - git-basics - Visual Studio Code'. The left sidebar is titled 'GIT-BASICS' and lists three files: 'initial-commit.txt', 'second-commit.txt', and 'Working\_with\_Branches.txt'. The 'TERMINAL' tab is selected at the top, showing the command line history:

```
D:\one drive data\Desktop\git-basics>git add Working_with_Branches.txt
D:\one drive data\Desktop\git-basics>git commit -m " Branches are Working"
[quickfix cd8816c] Branches are Working
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Working_with_Branches.txt

D:\one drive data\Desktop\git-basics>
```

Fig. Working with Branches

```
D:\one drive data\Desktop\git-basics>git log
commit cd8816cdbe74cecbaf4f5995e25b2f9bab3cb9a3 (HEAD -> quickfix)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Thu May 19 15:44:16 2022 +0530

    Branches are Working

commit dc4a67141cb3c3529cdc7da6200e9600f0c56387 (master, Landing-Page)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Thu May 19 15:02:23 2022 +0530

    added second-commit.txt

commit 320726669068faa962f8e245df7e3be52c76accc
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Thu May 19 02:36:24 2022 +0530

    updated file.txt
```

Fig. Heading with third branch

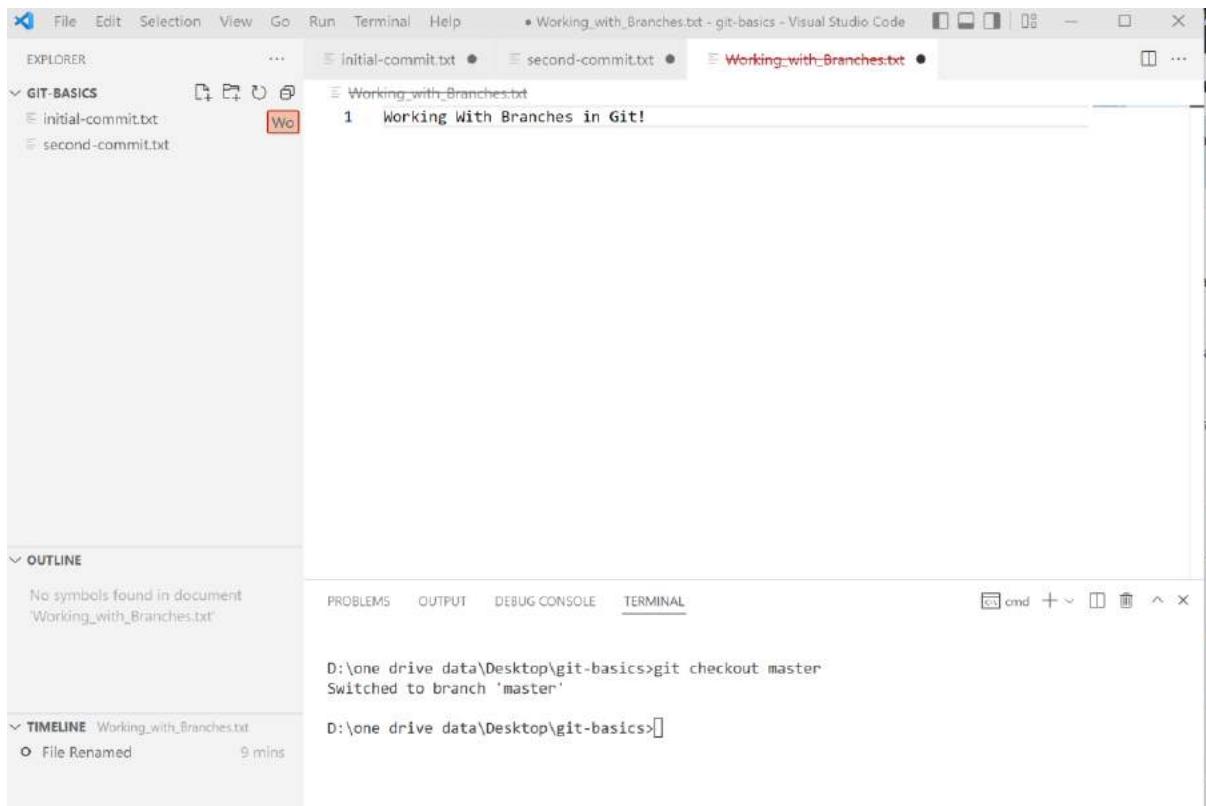


Fig. Switch to Master Branch

**After Switch to master branch? Does it reflect the Working with Branches in third branch (quickfix)? How to merge this branches ??**

## Merging Branches – The Basics

D:\one drive data\Desktop\git-basics>git merge quickfix

The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Terminal:** Shows the command "D:\one drive data\Desktop\git-basics>git merge quickfix". The output shows:
  - Landing-Page
  - \* master
  - quickfix
- Explorer:** Shows three files: initial-commit.txt, second-commit.txt, and Working\_with\_Branches.txt (which is currently selected).
- Outline:** Shows "No symbols found in document 'Working\_with\_Branches.txt'".
- Timeline:** Shows two events:
  - Branches are Working 5ai... 10 mins ago
  - File Renamed 13 mins ago
- Bottom Status Bar:** Shows "master" and other status indicators.

Fig. Merging with Branch

```

File Edit Selection View Go Run Terminal Help * Working_with_Branches.txt - git-basics - Visual Studio Code
EXPLORER ... PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL cmd + v ⌘ v X
GIT-BASICS
initial-commit.txt
second-commit.txt
Working_with_Branches.txt
D:\one drive data\Desktop\git-basics>git log
commit cd8816cdbe74cecbaf4f5995e25b2f9bab3cb9a3 (HEAD -> master, quickfix)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date: Thu May 19 15:44:16 2022 +0530

    Branches are Working

commit dc4a67141cb3c3529cdc7da6200e9600f0c56387 (Landing-Page)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date: Thu May 19 15:02:23 2022 +0530

    added second-commit.txt

commit 320726669068faa962f8e245df7e3be52c76accc
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date: Thu May 19 02:36:24 2022 +0530

    updated file.txt

D:\one drive data\Desktop\git-basics>

```

OUTLINE

No symbols found in document "Working\_with\_Branches.txt"

TIMELINE Working\_with\_Branches.txt

- Branches are Working Sai... 10 mins
- File Renamed 13 mins

Fig. Success History of Merging.

## Understanding the HEAD

The Latest commit is known as HEAD.

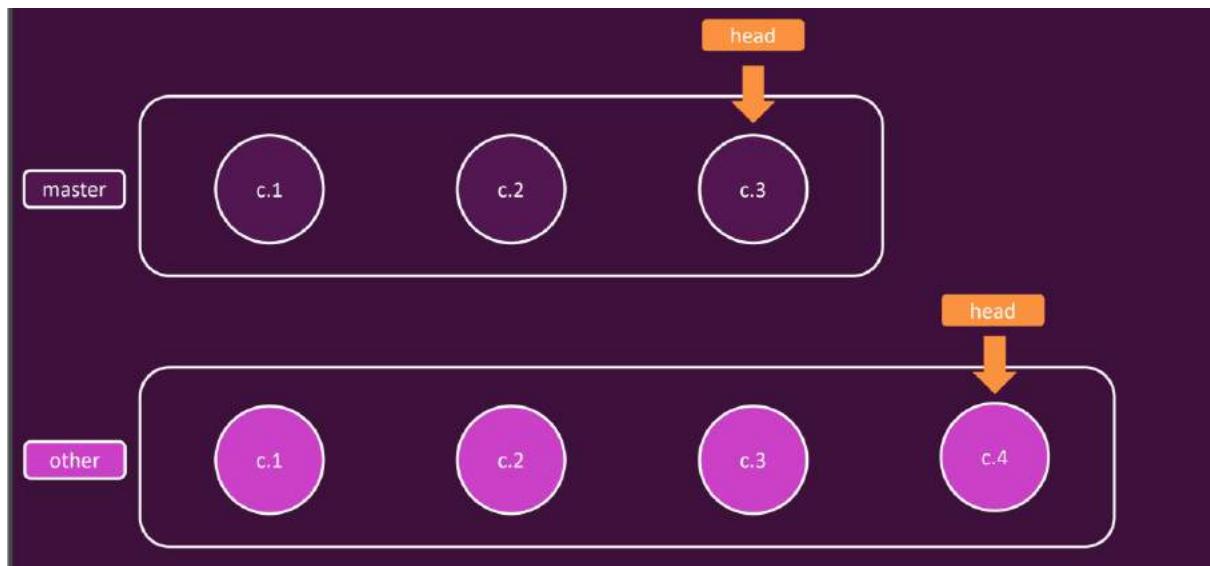


Fig. Head Understanding

The screenshot shows the Visual Studio Code interface with the terminal tab selected. The terminal window displays the following git session:

```
D:\one drive data\Desktop\git-basics>git branch
  * master
    quickfix

D:\one drive data\Desktop\git-basics>git checkout Landing-Page
Switched to branch 'Landing-Page'

D:\one drive data\Desktop\git-basics>git log
commit dc4a67141cb3c3529cdc7da6200e9600f0c56387 (HEAD -> Landing-Page)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Thu May 19 15:02:23 2022 +0530

    added second-commit.txt

commit 320726669068faa962f8e245df7e3be52c76accc
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Thu May 19 02:36:24 2022 +0530

    updated file.txt

D:\one drive data\Desktop\git-basics>
```

Fig. Head Understanding History

The screenshot shows the Visual Studio Code interface with the terminal tab selected. The terminal window displays the same git session as the previous screenshot. Additionally, the timeline sidebar is expanded, showing the following activity:

- Branches are Working
- commit dc4a67141cb3c3529cdc7da6200e9600f0c56387 (Landing-Page)
- commit 320726669068faa962f8e245df7e3be52c76accc
- updated file.txt

Fig. Head Understanding with Latest Commit on History

## Detached Head

### What is Detached Head?

Detached HEAD indicates that the currently checked-out repository is not a local branch. This can be caused by the following scenarios:

- When a branch is a read-only branch and we try to create a commit to that branch, then the commits can be termed as “free-floating” commits not connected to any branch. They would be in a detached state.
- When we checkout a tag or a specific commit and then we try to perform a new commit, then again the commits would not be connected to any branch. When we now try to checkout a branch, these new commits would be automatically placed at the top

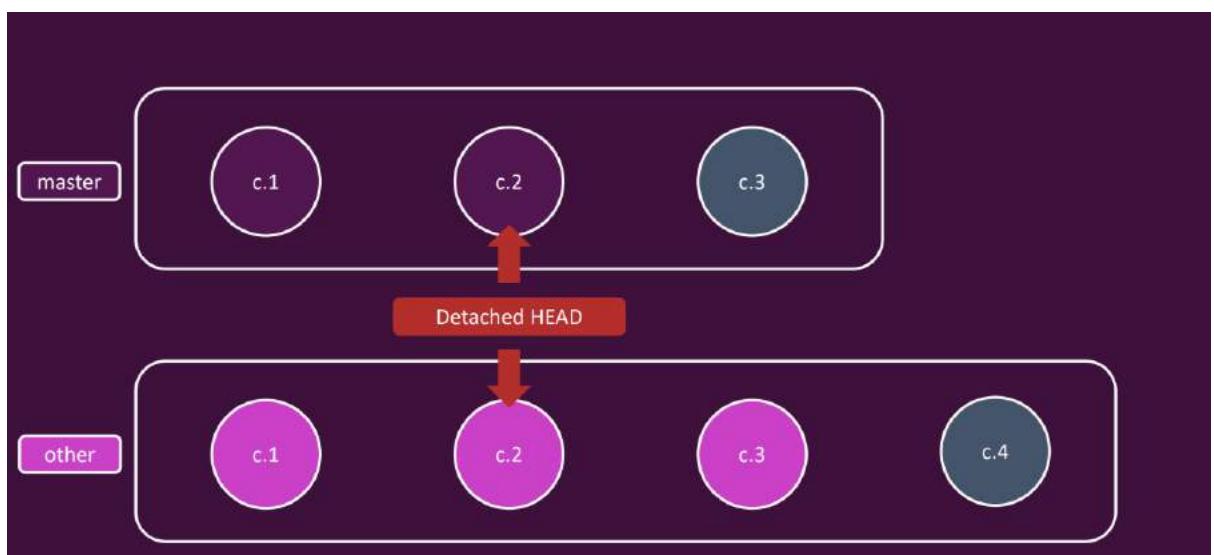


Fig. Detached Head

The screenshot shows the Visual Studio Code interface with the 'TERMINAL' tab selected. The terminal window displays the following output:

```

commit dc4a67141cb3c3529cdc7da6200e9600f0c56387 (Landing-Page)
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date: Thu May 19 15:02:23 2022 +0530

    added second-commit.txt

commit 320726669068faa962f8e245df7e3be52c76accc
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date: Thu May 19 02:36:24 2022 +0530

    updated file.txt

D:\one drive data\Desktop\git-basics>git checkout cd8816cdbe74cecbaf4f5995e25b2f9bab3cb9a3
Note: switching to 'cd8816cdbe74cecbaf4f5995e25b2f9bab3cb9a3'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at cd8816c Branches are Working

D:\one drive data\Desktop\git-basics>git branch
* (HEAD detached at cd8816c)
  Landing-Page
  master
  quickfix

```

The terminal window title is 'Working\_with\_Branches.txt - git-basics - Visual Studio Code'. The status bar at the bottom shows 'Ln 1, Col 30 Spaces:4 UTF-8 CRLF Plain Text'.

Fig. Detached Head on terminal

## How to avoid this?

In order to ensure that detached state doesn't happen, instead of checking out commit/tag, we can create a branch emanating from that commit and then we can switch to that newly created branch by using the command:

**git checkout <<branch\_name>>**. This ensures that a new branch is checked out and not a commit/tag thereby ensuring that a detached state wouldn't happen

The screenshot shows a Visual Studio Code interface with the following details:

- EXPLORER** panel: Shows files `initial-commit.txt`, `second-commit.txt`, and `Working_with_Branches.txt`.
- TERMINAL** tab: Active, showing a command-line session:

```
D:\one drive data\Desktop\git-basics>git branch
* (HEAD detached at cd8816c)
  Landing-Page
  master
  quickfix

D:\one drive data\Desktop\git-basics>git checkout quickfix
Switched to branch 'quickfix'

D:\one drive data\Desktop\git-basics>git branch
  Landing-Page
  master
* quickfix

D:\one drive data\Desktop\git-basics>
```
- OUTLINE** panel: Shows "No symbols found in document 'Working\_with\_Branches.txt'".
- TIMELINE** panel: Shows activity log:
  - Branches are Working Saif P... 2 hrs ago
  - File Renamed
- Bottom Status Bar**: Shows file path `Working_with_Branches.txt`, line 1, column 30, spaces: 4, encoding: UTF-8, CRLF, Plain Text, and icons for file operations.

Fig. Avoided Detached Head

## Branches & “git switch” (Git 2.2.23)

### The “git switch” command

The switch command allows you to do , well, switch branches and create new branches. Now you would say, "Why would I want to have a new command in here?" Well the idea basically is that checkout can be used for commits and for branches so it can be confusing, especially for beginners. So with switch is the nice shortcut for creating a new branch.

The screenshot shows the Visual Studio Code interface with the 'TERMINAL' tab selected. The terminal window displays the following command-line session:

```
D:\one drive data\Desktop\git-basics>git branch
  Landing-Page
* master
* quickfix

D:\one drive data\Desktop\git-basics>git switch Landing-Page
Switched to branch 'Landing-Page'

D:\one drive data\Desktop\git-basics>git switch -c commitsa
Switched to a new branch 'commitsa'

D:\one drive data\Desktop\git-basics>git branch
  Landing-Page
* commitsa
* master
* quickfix

D:\one drive data\Desktop\git-basics>
```

The 'EXPLORER' sidebar shows three files: 'initial-commit.txt', 'second-commit.txt', and 'Working\_with\_Branches.txt'. The 'OUTLINE' sidebar indicates 'No symbols found in document 'Working\_with\_Branches.txt''.

Fig. Created new branch with git switch

### How to reserve the Steps and fix when something wrong with working directory, Branches & Commit:

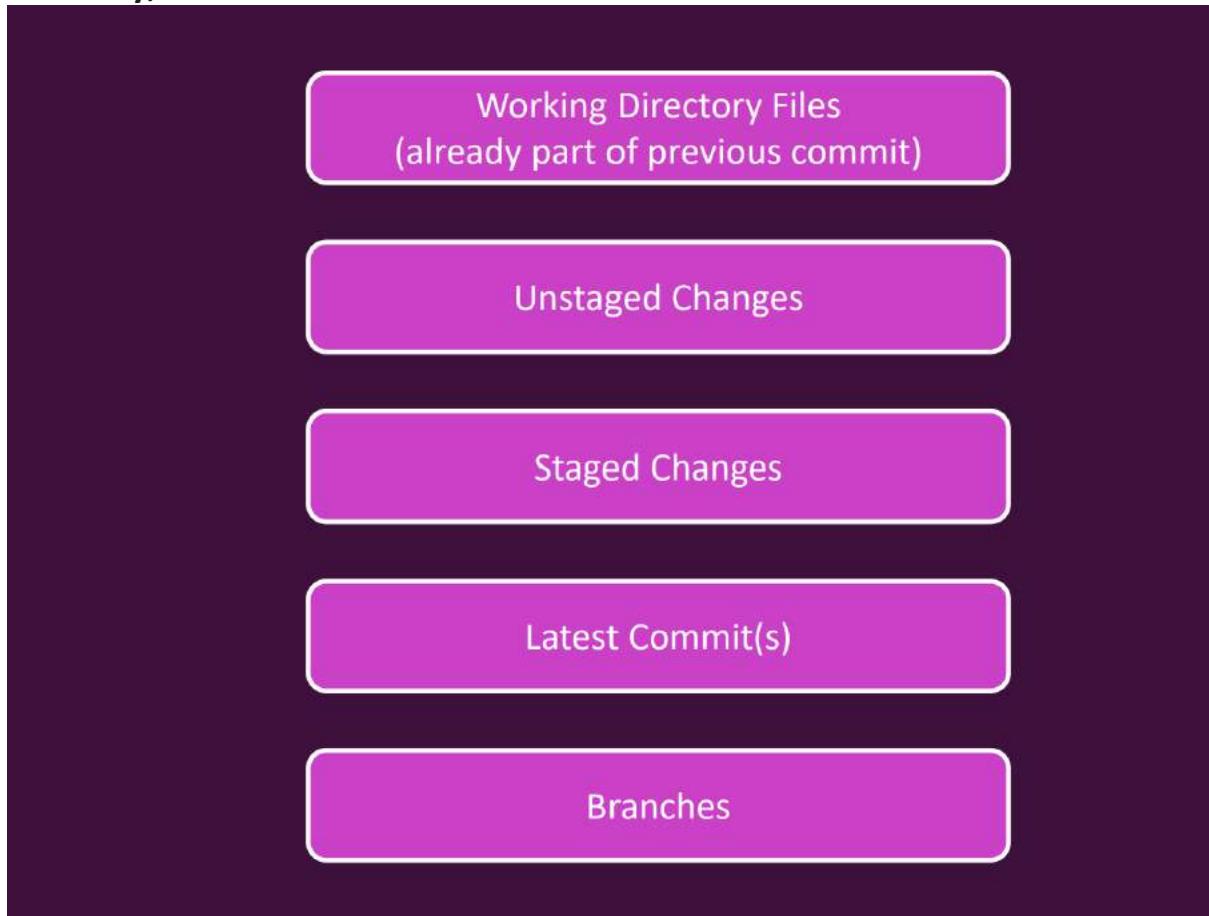
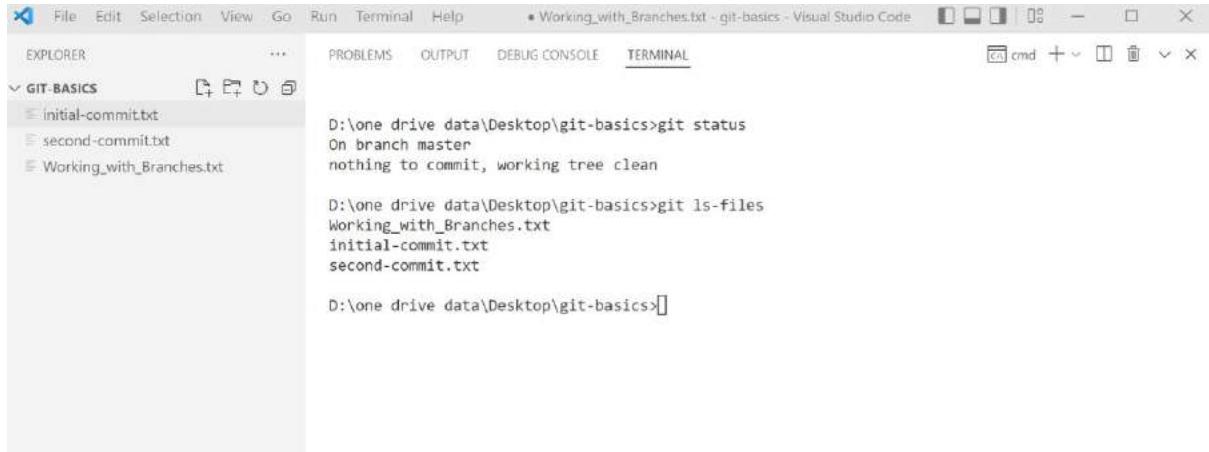


Fig. Deleting Data : Overview

## Deleting Working Directory Files:

To check Which files currently on staging area.

**git ls-files**

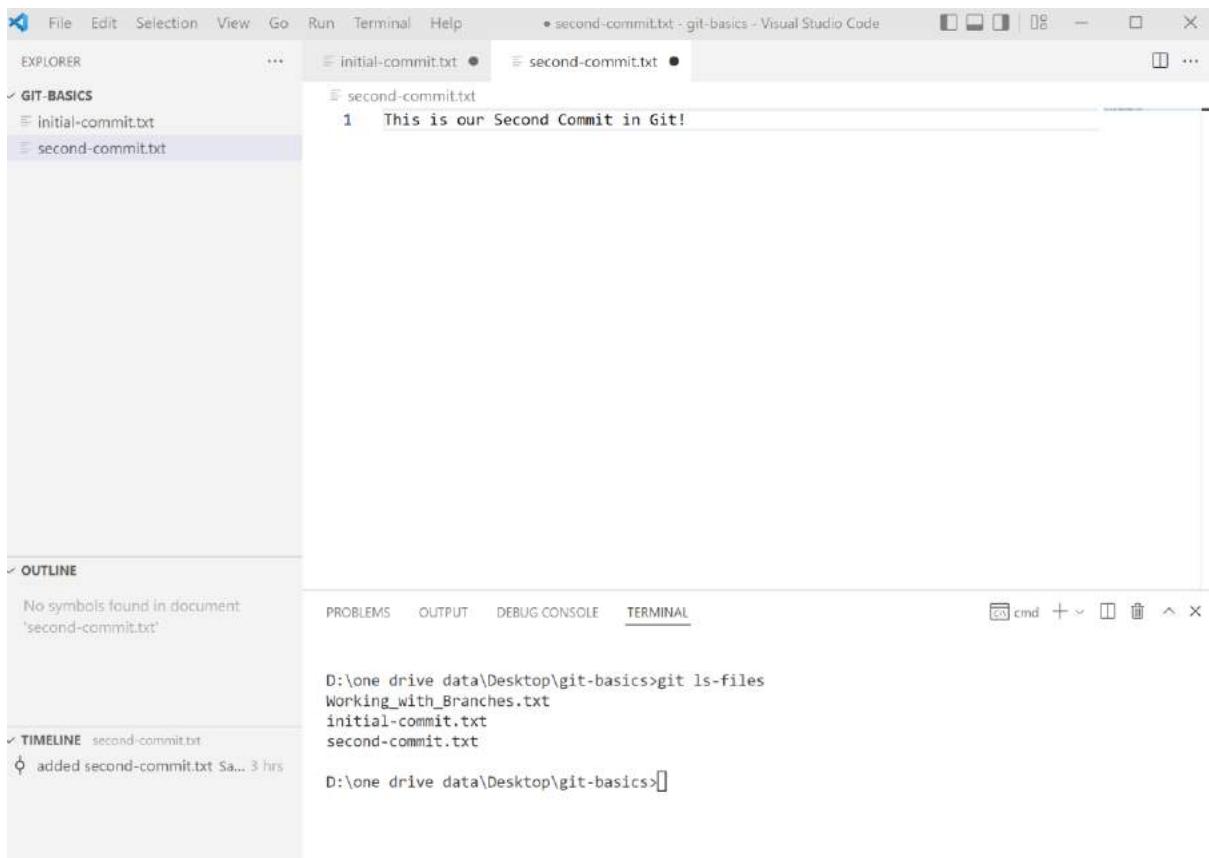


```
D:\one drive data\Desktop\git-basics>git status
On branch master
nothing to commit, working tree clean

D:\one drive data\Desktop\git-basics>git ls-files
Working_with_Branches.txt
initial-commit.txt
second-commit.txt

D:\one drive data\Desktop\git-basics>
```

Fig. show the staging area of master branch master



```
D:\one drive data\Desktop\git-basics>git ls-files
Working_with_Branches.txt
initial-commit.txt
second-commit.txt

D:\one drive data\Desktop\git-basics>
```

Fig. After deletion of file still show on staging area.

## How to remove the file which has been deleted from working directory? In staging area??

The screenshot shows the Visual Studio Code interface with the Terminal tab selected. The terminal window displays the following command-line session:

```
D:\one drive data\Desktop\git-basics>git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    deleted:   Working_with_Branches.txt

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

D:\one drive data\Desktop\git-basics>git rm Working_with_Branches.txt
rm 'Working_with_Branches.txt'

D:\one drive data\Desktop\git-basics>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    deleted:   Working_with_Branches.txt

D:\one drive data\Desktop\git-basics>git ls-files
initial-commit.txt
second-commit.txt

D:\one drive data\Desktop\git-basics>git commit -m "Deletion of Working_with_Branches.txt"
[master ccf3569] Deletion of Working_with_Branches.txt
  1 file changed, 0 insertions(+), 0 deletions(-)
  delete mode 100644 Working_with_Branches.txt

D:\one drive data\Desktop\git-basics>
```

The terminal output shows the steps to delete a file from the staging area: `git rm` followed by `git commit`.

Fig. Delete successful from staging area

## Undoing Unstaged Changes?

Added Extra Information not needed on Initial -Commit .txt

The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Explorer:** Shows a folder named "GIT-BASICS" containing "initial-commit.txt" and "second-commit.txt".
- Editor:** Two tabs are open: "initial-commit.txt" and "second-commit.txt". The content of "initial-commit.txt" is:

```
1 Our First Step in git!
2
3 The text is no longer needed!
```
- Terminal:** Shows the command line output of "git status":

```
D:\one drive data\Desktop\git-basics>git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   initial-commit.txt

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

D:\one drive data\Desktop\git-basics>
```
- Bottom Status Bar:** Shows "master\*", "0 0 △ 0", "Ln 3, Col 30", "Spaces: 4", "UTF-8", "CRLF", "Plain Text", and icons for file operations.

Fig. Unstaged Changes

## How to go back from these changes??

**git checkout initial-commit.txt**

```
D:\one drive data\Desktop\git-basics>git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   initial-commit.txt

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

D:\one drive data\Desktop\git-basics>git checkout intial-commit.txt
error: pathspec 'intial-commit.txt' did not match any file(s) known to git

D:\one drive data\Desktop\git-basics>git checkout initial-commit.txt
Updated 1 path from the index

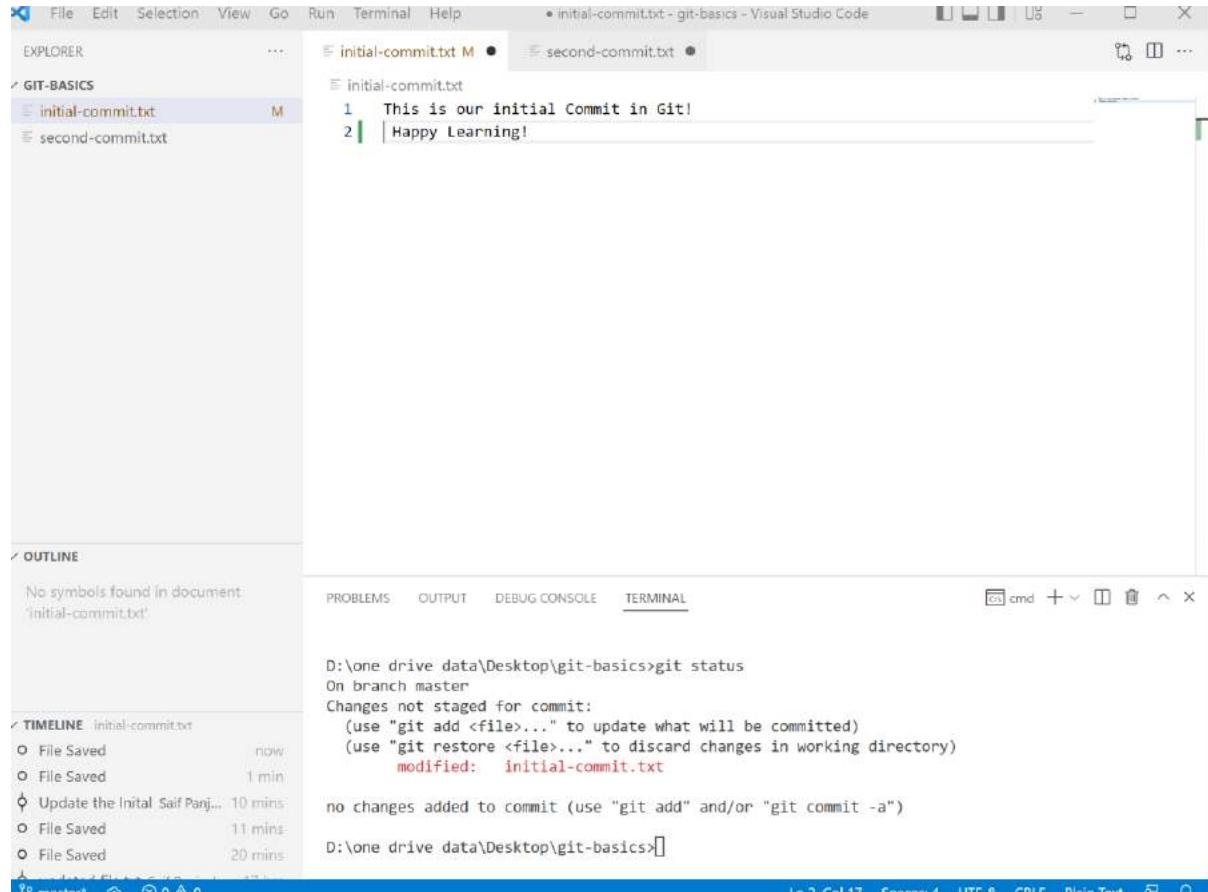
D:\one drive data\Desktop\git-basics>git status
On branch master
nothing to commit, working tree clean

D:\one drive data\Desktop\git-basics>
```

Fig. Undoing Unstaged Changes

## Latest Command for Undoing Unstaged Changes After (Git 2.2.23)

**git restore**: latest command for Undoing Unstaged Changes after git 2.2.23



The screenshot shows the Visual Studio Code interface. In the Explorer sidebar, there are two files: 'initial-commit.txt' and 'second-commit.txt'. The 'initial-commit.txt' file is selected. In the main editor area, the content of 'initial-commit.txt' is displayed:

```
1 This is our initial Commit in Git!
2 | Happy Learning!
```

In the Timeline sidebar, it shows a history of changes:

- File Saved now
- File Saved 1 min
- Update the initial Salif Panj... 10 mins
- File Saved 11 mins
- File Saved 20 mins

The terminal tab shows the command line output:

```
D:\one drive data\Desktop\git-basics>git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
    (use "git restore <file>..." to discard changes in working directory)
      modified:   initial-commit.txt

no changes added to commit (use "git add" and/or "git commit -a")
D:\one drive data\Desktop\git-basics>
```

Fig. Unstaged Changes Happens



The screenshot shows the Visual Studio Code interface with the terminal tab active. The command line output is:

```
D:\one drive data\Desktop\git-basics>git restore initial-commit.txt
D:\one drive data\Desktop\git-basics>git status
On branch master
nothing to commit, working tree clean

D:\one drive data\Desktop\git-basics>
```

Fig. Successful restore back the changes

The screenshot shows the Visual Studio Code interface with the following details:

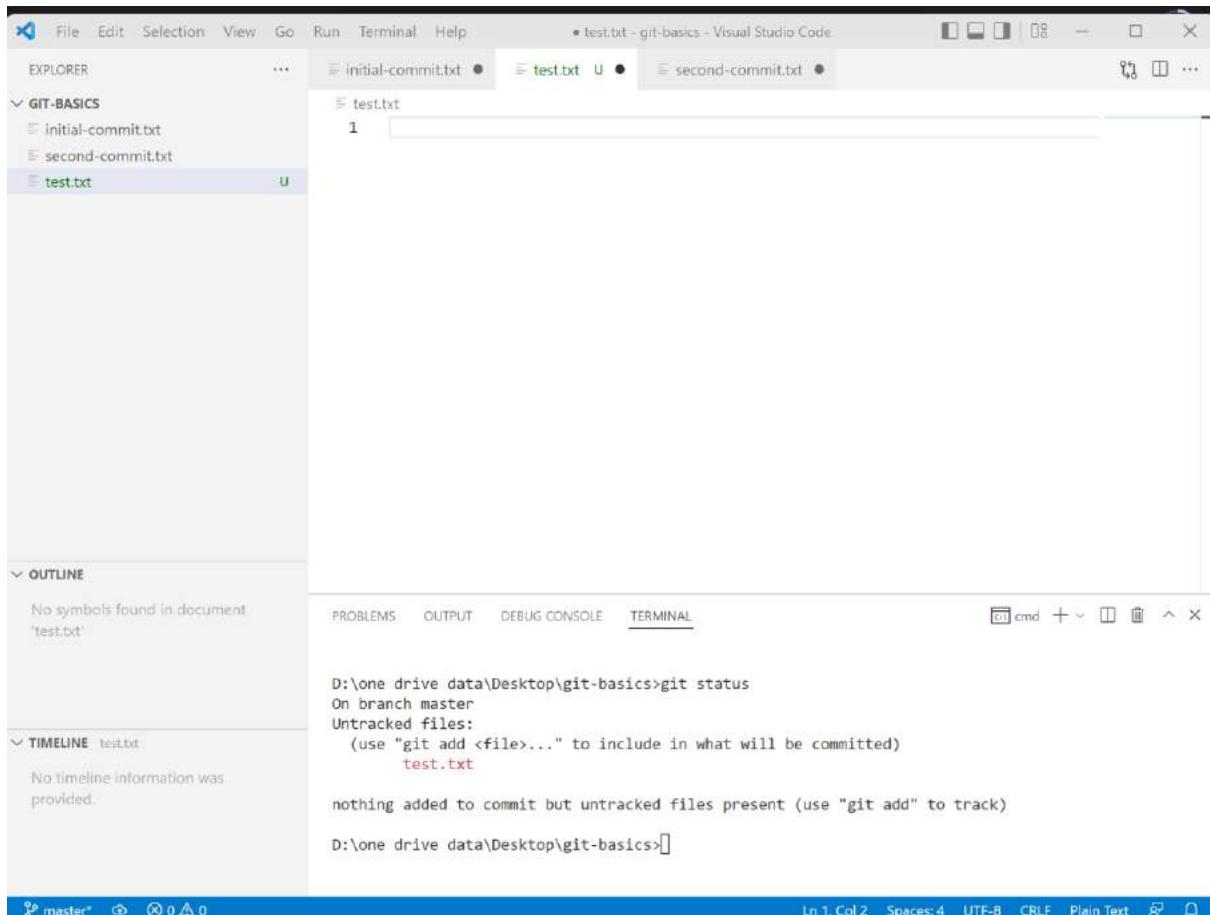
- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Explorer:** Shows a folder named "GIT-BASICS" containing two files: "initial-commit.txt" and "second-commit.txt".
- Editor:** Two tabs are open: "initial-commit.txt" and "second-commit.txt". The "initial-commit.txt" tab contains the text "1 This is our initial Commit in Git!" and "2".
- Terminal:** The terminal window shows the command line history:

```
D:\one drive data\Desktop\git-basics>git restore .
D:\one drive data\Desktop\git-basics>git status
On branch master
nothing to commit, working tree clean
D:\one drive data\Desktop\git-basics>[]
```
- Timeline:** A timeline for "initial-commit.txt" is shown, listing several file save operations over a 23-minute period.
- Bottom Status Bar:** Shows "In 2 Col 1 Spaces 4 HTAB 8 CR/LF Plain Text".

Fig. Successful restore back the changes on multiple statement

## How to restore Unstaged File:

“git clean -dn” & “git clean -df”: - remove files or remove force file



The screenshot shows the Visual Studio Code interface. In the Explorer panel, there are three files: 'initial-commit.txt', 'second-commit.txt', and 'test.txt'. 'test.txt' is highlighted and has a 'U' icon next to it, indicating it is unstaged. The terminal tab is active, displaying the following command and output:

```
D:\one drive data\Desktop\git-basics>git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    test.txt

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

D:\one drive data\Desktop\git-basics>
```

The status bar at the bottom shows 'Ln 1, Col 2' and other file details.

Fig Created file test.txt



The screenshot shows the Visual Studio Code interface. In the Explorer panel, 'initial-commit.txt' and 'second-commit.txt' are selected. The terminal tab is active, displaying the following command and output:

```
D:\one drive data\Desktop\git-basics>git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    test.txt

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

D:\one drive data\Desktop\git-basics>git clean -dn
Would remove test.txt

D:\one drive data\Desktop\git-basics>git clean -df
Removing test.txt

D:\one drive data\Desktop\git-basics>
```

Fig. Remove test.txt

## Undoing Staged Changes:

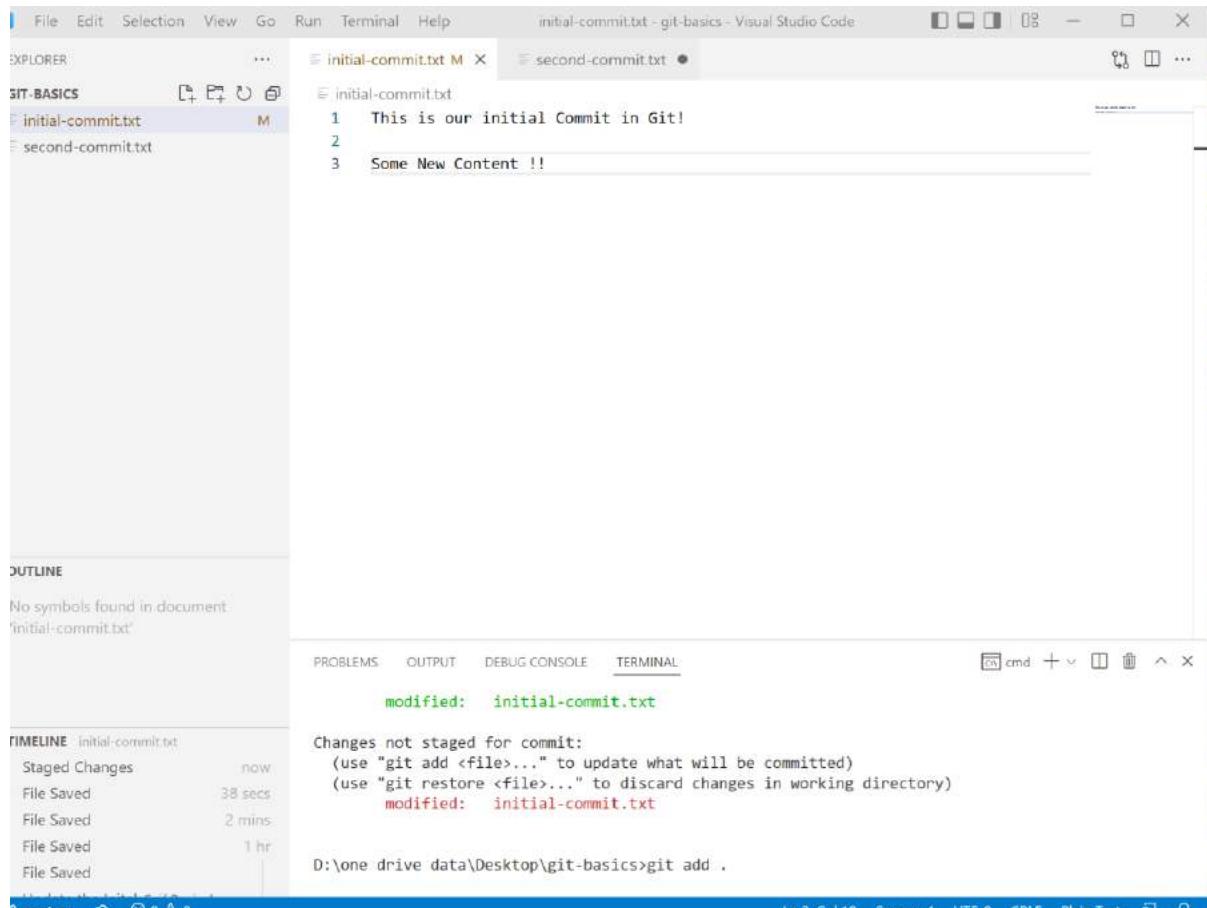


Fig. Added Some Content

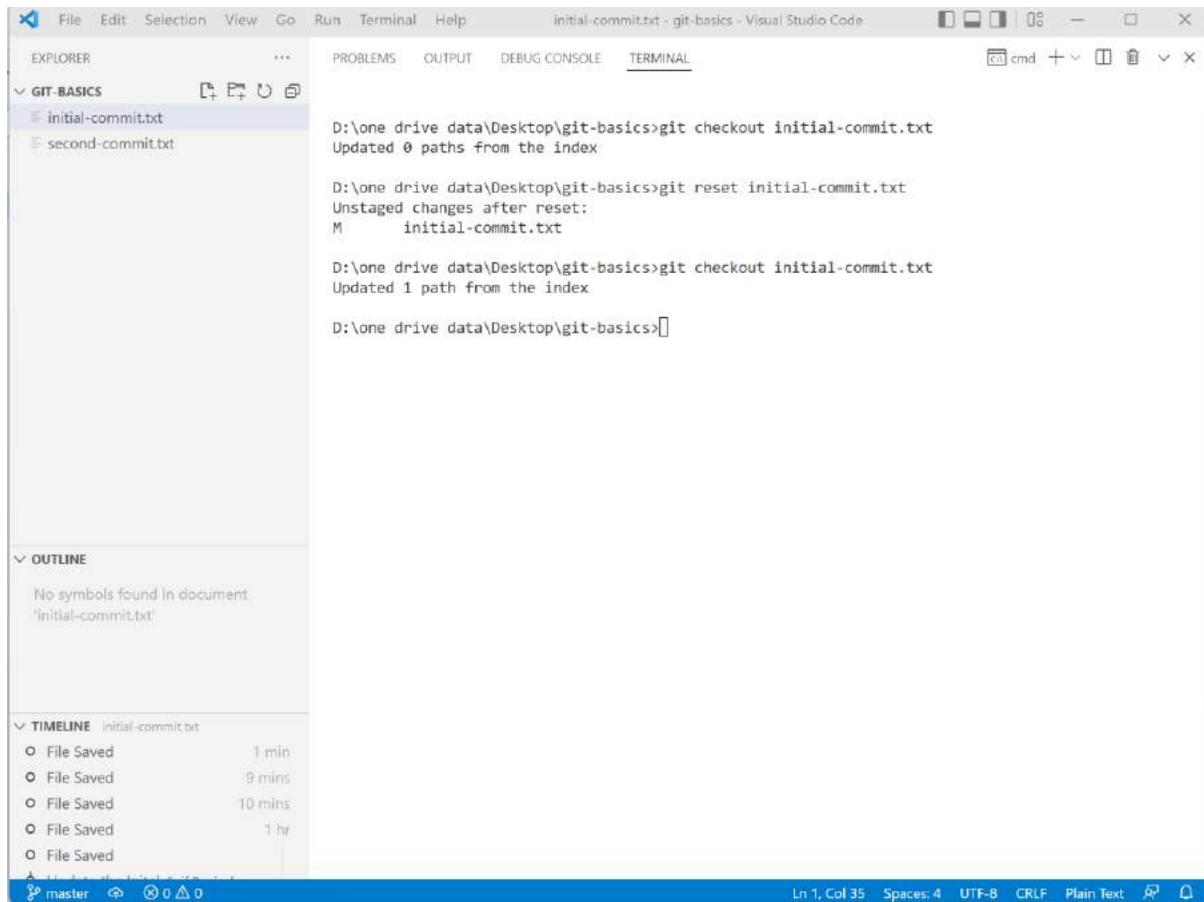
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

```
D:\one drive data\Desktop\git-basics>git status
On branch master
Changes to be committed:
(use "git restore --staged <file>..." to unstage)
modified: initial-commit.txt
```

```
D:\one drive data\Desktop\git-basics>
```

Fig. Modified git status

**git reset: will help you to bring latest status of commit in staging area and later we can undo the staged changes.**



The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help
- Terminal Tab:** initial-commit.txt - git-basics - Visual Studio Code
- Terminal Output:**

```
D:\one drive data\Desktop\git-basics>git checkout initial-commit.txt
Updated 0 paths from the index

D:\one drive data\Desktop\git-basics>git reset initial-commit.txt
Unstaged changes after reset:
M      initial-commit.txt

D:\one drive data\Desktop\git-basics>git checkout initial-commit.txt
Updated 1 path from the index

D:\one drive data\Desktop\git-basics>
```
- Explorer View:** Shows files: initial-commit.txt, second-commit.txt
- Outline View:** No symbols found in document 'initial-commit.txt'
- Timeline View:** Shows five 'File Saved' events with timestamps: 1 min, 9 mins, 10 mins, 1 hr, and a recent one.
- Bottom Status Bar:** master, 0 △ 0, Ln 1, Col 35, Spaces: 4, UTF-8, CRLF, Plain Text

Fig. Staging Changes reset

## git restore: will act as same as “git reset” after version Git 2.2.23

```
D:\One Drive\DATA\Desktop\git-basics>git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
    (use "git restore <file>..." to discard changes in working directory)
      modified:   second-commit.txt

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

D:\One Drive\DATA\Desktop\git-basics>git add .

D:\One Drive\DATA\Desktop\git-basics>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   second-commit.txt

D:\One Drive\DATA\Desktop\git-basics>
```

```
D:\One Drive\DATA\Desktop\git-basics>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   second-commit.txt

D:\One Drive\DATA\Desktop\git-basics>git restore --staged second-commit.txt
D:\One Drive\DATA\Desktop\git-basics>git checkout second-commit.txt
Updated 1 path from the index

D:\One Drive\DATA\Desktop\git-basics>git status
On branch master
nothing to commit, working tree clean

D:\One Drive\DATA\Desktop\git-basics>
```

Fig. Success Staged Changes restore

## Deleting commit with git reset

The screenshot shows two instances of Visual Studio Code side-by-side, illustrating the steps to delete a commit from the master branch.

**Top Window (Initial State):**

- Explorer:** Shows three files: `initial-commit.txt`, `second-commit.txt`, and `unrequired.txt`.
- Terminal:** Shows the command history:
  - `D:\one drive data\Desktop\git-basics>git add unrequired.txt`
  - `D:\one drive data\Desktop\git-basics>git ls-files`  
initial-commit.txt  
second-commit.txt  
unrequired.txt
  - `D:\one drive data\Desktop\git-basics>git commit -m "Unrequired File"`  
[master f4fd19c] Unrequired File  
1 file changed, 0 insertions(+), 0 deletions(-)  
create mode 100644 unrequired.txt
  - `D:\one drive data\Desktop\git-basics>`
- Timeline:** Shows a single entry: "Unrequired File Saif Panjesha now".

**Bottom Window (After Reset):**

- Explorer:** Shows the same three files: `initial-commit.txt`, `second-commit.txt`, and `unrequired.txt`.
- Terminal:** Shows the command history:
  - `D:\one drive data\Desktop\git-basics>git log`  
commit be580d26517e57dd716627f1acfe5908fcbb0d3a (HEAD -> master)  
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>  
Date: Thu May 19 22:20:15 2022 +0530
  - `unrequired.txt`
  - `commit f4fd19c066db84e9502c941d12ae898b3997696d`  
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>  
Date: Thu May 19 22:09:48 2022 +0530
  - `Unrequired File`
  - `commit 26eccace006f791648def83186b61533289a157ac`  
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>  
Date: Thu May 19 20:16:58 2022 +0530
  - `Update the Initial`
  - `commit ccf3569c84dbc98f730a736a8aba7e8ff39d16af`  
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>  
Date: Thu May 19 18:31:46 2022 +0530

Fig. Logs Heading at Master staging area

As fig show I want to go previous head the update the Initial Commit done by “git reset” with “- -soft” as soft reset

D:\one drive data\Desktop\git-basics>git reset --soft HEAD~2

The screenshot shows the Visual Studio Code interface with the terminal tab active. The terminal window displays the command "D:\one drive data\Desktop\git-basics>git reset --soft HEAD~2" followed by the output of the "git log" command. The log shows three commits:

```
D:\one drive data\Desktop\git-basics>git log
commit 26eccace006f791648def83186b615f3289a157ac (HEAD -> master)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Thu May 19 20:16:58 2022 +0530

    Update the Initial

commit ccf3569c84dbc98f730a736a8aba7e8ff39d16af
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Thu May 19 18:31:46 2022 +0530

    Deletion of Working_with_Branches.txt

commit cd8816cdbe74cecbaf4f5995e25b2f9bab3cb9a3 (quickfix, commitsa, Landing-Page)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Thu May 19 15:44:16 2022 +0530

    Branches are Working

commit dc4a67141cb3c3529cdc7da6200e9600f0c56387
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Thu May 19 15:02:23 2022 +0530

    added second-commit.txt

commit 320726669068faa962f8e245df7e3be52c76accc
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Thu May 19 02:36:24 2022 +0530
```

The Explorer sidebar shows files: initial-commit.txt, second-commit.txt, and unrequired.txt. The Timeline sidebar shows staged changes and file saves.

Fig. Head -> master to Update the Initial state

## Default Command: git reset HEAD ~2

The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Tab Bar:** unrequired.txt - git-basics - Visual Studio Code.
- Explorer:** Shows a folder named "GIT-BASICS" containing three files: "initial-commit.txt", "second-commit.txt", and "unrequired.txt".
- Terminal:** Shows the command "git reset HEAD~2" being run in the terminal. The output indicates that after the reset, there are unstaged changes, specifically the file "initial-commit.txt". It then displays the log for the current branch, which includes two commits: one for "second-commit.txt" and another for "updated file.txt".
- Outline:** Shows that no symbols were found in the document "unrequired.txt".
- Timeline:** Shows a single entry: "File Saved" 12 mins ago.

Fig. Head -> master to added second-commit.txt

## Successful Deleting file:

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER**: Shows three files: initial-commit.txt (M), second-commit.txt (M), and unrequired.txt (U).
- TERMINAL**: Displays the command `git reset HEAD~2` followed by the output:

```
D:\one drive data\Desktop\git-basics>git reset HEAD~2
Unstaged changes after reset:
M      initial-commit.txt

D:\one drive data\Desktop\git-basics>git log
commit dc4a67141cb3c3529cdc7da6200e9600f0c56387 (HEAD -> master)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Thu May 19 15:02:23 2022 +0530

    added second-commit.txt

commit 320726669068faa962f8e245df7e3be52c76accc
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Thu May 19 02:36:24 2022 +0530

    updated file.txt

D:\one drive data\Desktop\git-basics>
```
- OUTLINE**: Shows "No symbols found in document 'unrequired.txt'".
- TIMELINE**: Shows "File Saved" 12 mins ago.
- STATUS BAR**: Shows "master\*" and other status indicators.

Fig. Success Deletion on unrequired file

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER**: Shows three files: initial-commit.txt, second-commit.txt, and unrequired.txt (X).
- TERMINAL**: Displays the command `git ls-files` followed by the output:

```
D:\one drive data\Desktop\git-basics>git ls-files
initial-commit.txt
second-commit.txt
```
- EDITOR**: Shows the content of unrequired.txt: "1 Some content".
- STATUS BAR**: Shows "D:\one drive data\Desktop\git-basics>".

Fig. Added file again

"git reset" with "- -hard" as Hard reset

## Removing all changes from Working directory & Staging area

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER**: Shows files: initial-commit.txt, second-commit.txt, and unrequired.txt.
- GIT-BASICS**: Shows files: initial-commit.txt, second-commit.txt.
- OUTLINE**: No symbols found in document 'unrequired.txt'.
- TIMELINE**: File Saved 24 mins ago.
- TERMINAL**:

```
D:\one\drive\data\Desktop\git-basics>git reset --hard HEAD~1
HEAD is now at dc4a671 added second-commit.txt

D:\one\drive\data\Desktop\git-basics>git ls-files
initial-commit.txt
second-commit.txt

D:\one\drive\data\Desktop\git-basics>[]
```

Fig. -- hard removing all area

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER**: Shows files: initial-commit.txt, second-commit.txt.
- GIT-BASICS**: Shows files: initial-commit.txt, second-commit.txt.
- OUTLINE**: No symbols found in document 'unrequired.txt'.
- TIMELINE**: File Saved 24 mins ago.
- TERMINAL**:

```
D:\one\drive\data\Desktop\git-basics>git reset --hard HEAD~1
HEAD is now at dc4a671 added second-commit.txt

D:\one\drive\data\Desktop\git-basics>git ls-files
initial-commit.txt
second-commit.txt

D:\one\drive\data\Desktop\git-basics>git log
commit dc4a67141cb3c3529cdc7da6200e9600f0c56387 (HEAD -> master)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Thu May 19 15:02:23 2022 +0530

    added second-commit.txt

commit 320726669068faa962f8e245df7e3be52c76accc
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Thu May 19 02:36:24 2022 +0530

    updated file.txt

D:\one\drive\data\Desktop\git-basics>[]
```

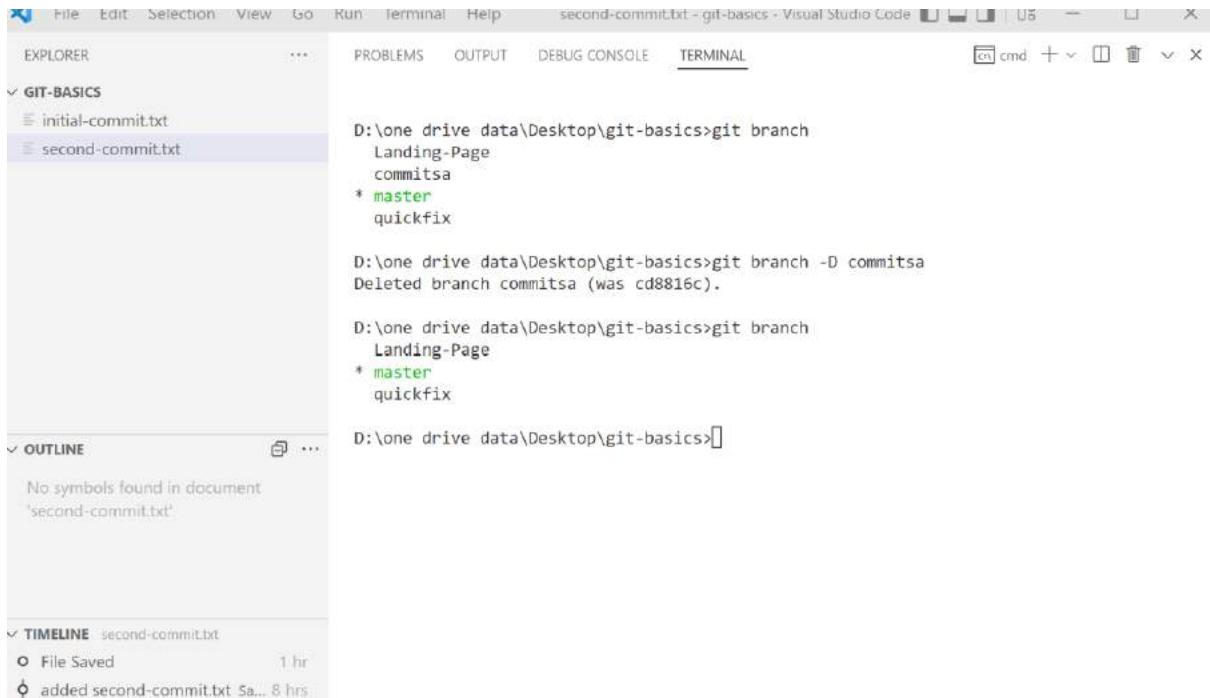
Fig. Head -> master to added second-commit.txt

## **Deleting Branches:**

**git branch -D <<branch-name>>:**

**-d: allows you to only delete the branches.**

**-D: allows you to force full delete your merge branches that not needed anymore**



The screenshot shows the Visual Studio Code interface with the 'second-commit.txt' file open. The terminal tab is active, displaying the following command and its execution:

```
D:\one\drive\data\Desktop\git-basics>git branch
  Landing-Page
  commitsa
* master
  quickfix

D:\one\drive\data\Desktop\git-basics>git branch -D commitsa
Deleted branch commitsa (was cd8816c).

D:\one\drive\data\Desktop\git-basics>
```

The Explorer sidebar shows two files: 'initial-commit.txt' and 'second-commit.txt'. The Timeline sidebar shows a file save event and the addition of 'second-commit.txt'.

**Fig. Commitsa Branch Deleted**

## **To delete multiple branches:**

```
D:\one\drive\data\Desktop\git-basics>git branch -D Landing-Page quickfix
Deleted branch Landing-Page (was cd8816c).
Deleted branch quickfix (was cd8816c).
```

```
D:\one\drive\data\Desktop\git-basics>
```

**Fig. Deleted multiple branches**

## Committing Detached Head Changes:

The Commit which is not part of any branches is called detached head state.

```
D:\one drive data\Desktop\git-basics>git branch
* master

D:\one drive data\Desktop\git-basics>git log
commit dc4a67141cb3c3529cdc7da6200e9600f0c56387 (HEAD -> master)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Thu May 19 15:02:23 2022 +0530

    added second-commit.txt

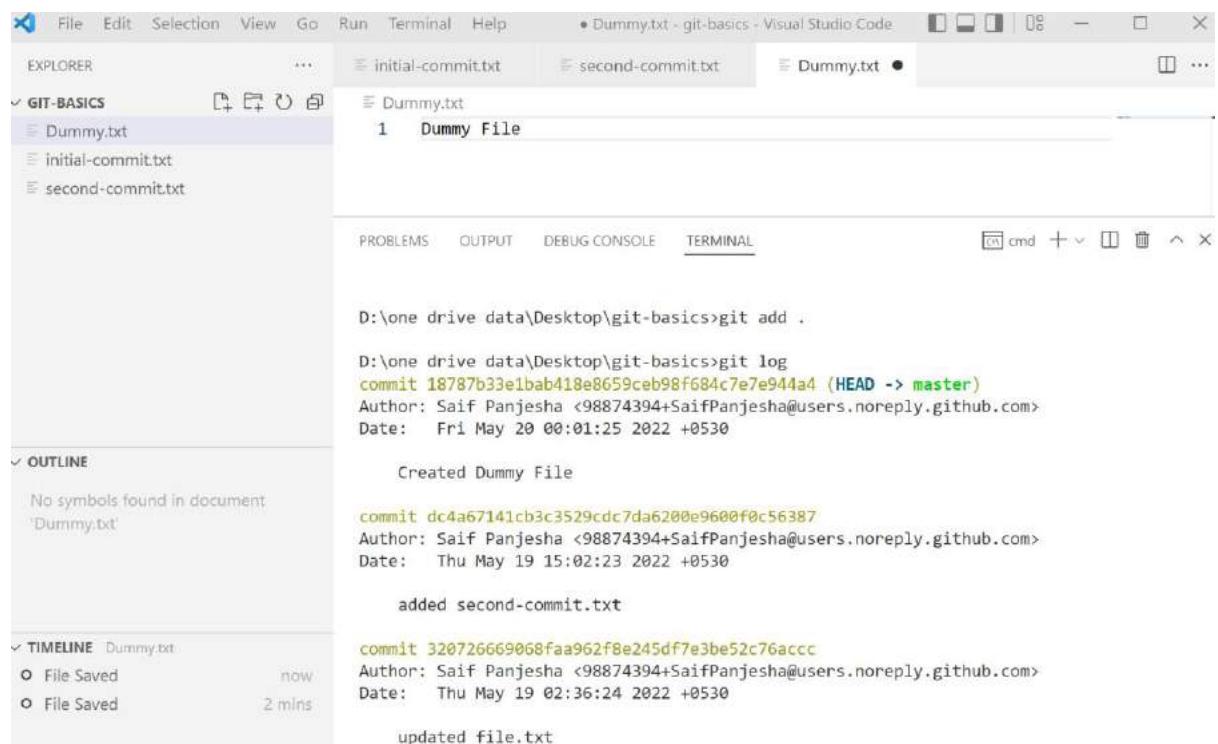
commit 320726669068faa962f8e245df7e3be52c76accc
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Thu May 19 02:36:24 2022 +0530

    updated file.txt

D:\one drive data\Desktop\git-basics>
```

Fig. Master branch with two commits.

## Creating a new File: dummy.txt



The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER:** Shows files: Dummy.txt, initial-commit.txt, second-commit.txt.
- TERMINAL:** Shows the command history:

```
D:\one drive data\Desktop\git-basics>git add .
D:\one drive data\Desktop\git-basics>git log
commit 18787b33e1bab418e8659ceb98f684c7e7e944a4 (HEAD -> master)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 00:01:25 2022 +0530

    Created Dummy File

commit dc4a67141cb3c3529cdc7da6200e9600f0c56387
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Thu May 19 15:02:23 2022 +0530

    added second-commit.txt

commit 320726669068faa962f8e245df7e3be52c76accc
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Thu May 19 02:36:24 2022 +0530

    updated file.txt
```
- OUTLINE:** Shows 'No symbols found in document 'Dummy.txt''.
- TIMELINE:** Shows events: 'File Saved' at 'now' and '2 mins' ago.

Fig. Dummy File

**Detached State:**

```
D:\one drive data\Desktop\git-basics>git checkout  
320726669068faa962f8e245df7e3be52c76accc
```

**Note: switching to '320726669068faa962f8e245df7e3be52c76accc'.**

**You are in 'detached HEAD' state. You can look around, make experimental changes and commit them, and you can discard any commits you make in this**

**state without impacting any branches by switching back to a branch.**

**If you want to create a new branch to retain commits you create, you may do so (now or later) by using -c with the switch command. Example:**

```
git switch -c <new-branch-name>
```

**Or undo this operation with:**

```
git switch -
```

**Turn off this advice by setting config variable advice.detachedHead to false**  
**HEAD is now at 3207266 updated file.txt**

```
D:\one drive data\Desktop\git-basics>git log  
commit 320726669068faa962f8e245df7e3be52c76accc (HEAD)  
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>  
Date: Thu May 19 02:36:24 2022 +0530
```

**updated file.txt**

```
D:\one drive data\Desktop\git-basics>git branch  
* (HEAD detached at 3207266)  
master
```

## **Got Unstaged or Untracked File**

```
D:\one drive data\Desktop\git-basics>git status
HEAD detached at 3207266
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified: initial-commit.txt
```

### **Untracked files:**

```
(use "git add <file>..." to include in what will be committed)
  detached-head.txt
```

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

---

```
D:\one drive data\Desktop\git-basics>git add .
```

```
D:\one drive data\Desktop\git-basics>git status
```

```
HEAD detached at 3207266
```

### **Changes not staged for commit:**

```
(use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified: initial-commit.txt
```

### **Untracked files:**

```
(use "git add <file>..." to include in what will be committed)
  detached-head.txt
```

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

```
D:\one drive data\Desktop\git-basics>git add .
```

```
D:\one drive data\Desktop\git-basics>git status
```

```
HEAD detached at 3207266
```

### **Changes to be committed:**

```
(use "git restore --staged <file>..." to unstage)
  new file: detached-head.txt
  modified: initial-commit.txt
```

---

Let's Commit Now:

```
D:\one drive data\Desktop\git-basics>git commit -m "Changes in detached head"
```

```
[detached HEAD b44e221] Changes in detached head
 2 files changed, 1 insertion(+)
 create mode 100644 detached-head.txt
```

```
D:\one drive data\Desktop\git-basics>git log
commit b44e221714e698dbb0f2d8364320407845546112 (HEAD)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date: Fri May 20 00:18:55 2022 +0530
```

Changes in detached head

```
commit 320726669068faa962f8e245df7e3be52c76accc
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date: Thu May 19 02:36:24 2022 +0530
```

updated file.txt

```
D:\one drive data\Desktop\git-basics>git branch
* (HEAD detached from 3207266)
  master
```

---

```
D:\one drive data\Desktop\git-basics>git switch master
```

Warning: you are leaving 1 commit behind, not connected to  
any of your branches:

b44e221 Changes in detached head

If you want to keep it by creating a new branch, this may be a good time  
to do so with:

```
git branch <new-branch-name> b44e221
```

Switched to branch 'master'

Detached branch gone:

```
D:\one drive data\Desktop\git-basics>git branch
* master
```

```
D:\one drive data\Desktop\git-basics>git branch detached-head b44e221
```

```
D:\one drive data\Desktop\git-basics>git branch  
detached-head  
* master
```

```
D:\one drive data\Desktop\git-basics>git checkout detached-head  
Switched to branch 'detached-head'
```

```
D:\one drive data\Desktop\git-basics>git switch master  
Switched to branch 'master'
```

```
D:\one drive data\Desktop\git-basics>git merge detached-head  
Merge made by the 'ort' strategy.  
detached-head.txt | 0  
initial-commit.txt | 1 +  
2 files changed, 1 insertion(+)  
create mode 100644 detached-head.txt
```

### Succesful Committing Detached Head Changes to master:

```
D:\one drive data\Desktop\git-basics>git ls-files  
detached-head.txt  
dummy.txt  
initial-commit.txt  
second-commit.txt
```

---

```
D:\one drive data\Desktop\git-basics>git log  
commit bd822cb7c7880499bafe6d035c08e6142bb9aa19 (HEAD -> master)  
Merge: 18787b3 b44e221  
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>  
Date: Fri May 20 00:30:11 2022 +0530
```

Merge branch 'detached-head'

commit b44e221714e698dbb0f2d8364320407845546112 (detached-head)  
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>  
Date: Fri May 20 00:18:55 2022 +0530

### Changes in detached head

commit 18787b33e1bab418e8659ceb98f684c7e7e944a4  
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>  
Date: Fri May 20 00:01:25 2022 +0530

### Created Dummy File

commit dc4a67141cb3c3529cdc7da6200e9600f0c56387  
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>  
Date: Thu May 19 15:02:23 2022 +0530

### added second-commit.txt

commit 320726669068faa962f8e245df7e3be52c76accc  
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>  
Date: Thu May 19 02:36:24 2022 +0530

### updated file.txt

D:\one drive data\Desktop\git-basics>git checkout  
dc4a67141cb3c3529cdc7da6200e9600f0c56387  
Note: switching to 'dc4a67141cb3c3529cdc7da6200e9600f0c56387'.

You are in 'detached HEAD' state. You can look around, make experimental changes and commit them, and you can discard any commits you make in this state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may do so (now or later) by using -c with the switch command. Example:

```
git switch -c <new-branch-name>
```

Or undo this operation with:

`git switch -`

Turn off this advice by setting config variable `advice.detachedHead` to false

HEAD is now at dc4a671 added second-commit.txt

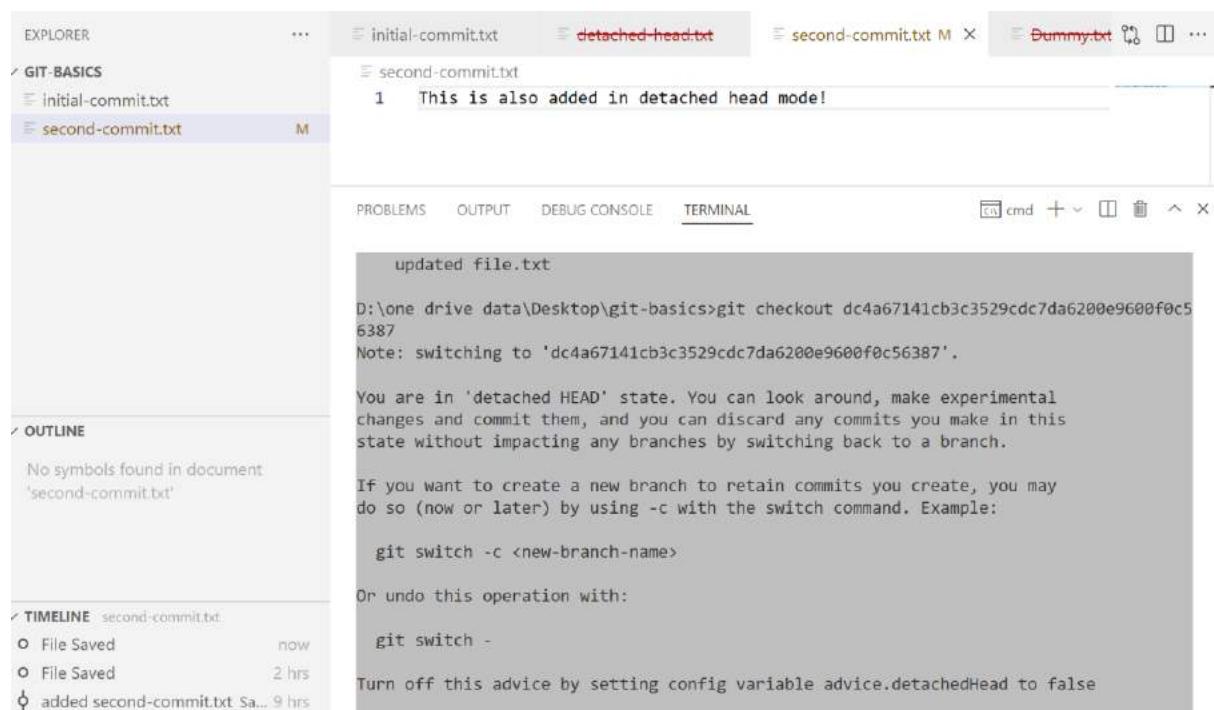


Fig. Added Some in Detached Head

```
D:\one drive data\Desktop\git-basics>git branch
* (HEAD detached at dc4a671)
  detached-head
    master

D:\one drive data\Desktop\git-basics>git add .

D:\one drive data\Desktop\git-basics>git commit -m "Text added to Detached Head"
[detached HEAD 4578010] Text added to Detached Head
 1 file changed, 1 insertion(+)

D:\one drive data\Desktop\git-basics>[]
```

Fig. Committed Save Changes

## Successful Committing Detached Head Changes to master:

```
D:\one drive data\Desktop\git-basics>git branch Detached2Head
```

```
D:\one drive data\Desktop\git-basics>git branch
```

```
* (HEAD detached from dc4a671)
```

```
Detached2Head
```

```
detached-head
```

```
master
```

```
D:\one drive data\Desktop\git-basics>git switch master
```

```
Previous HEAD position was 4578010 Text added to Detached Head
```

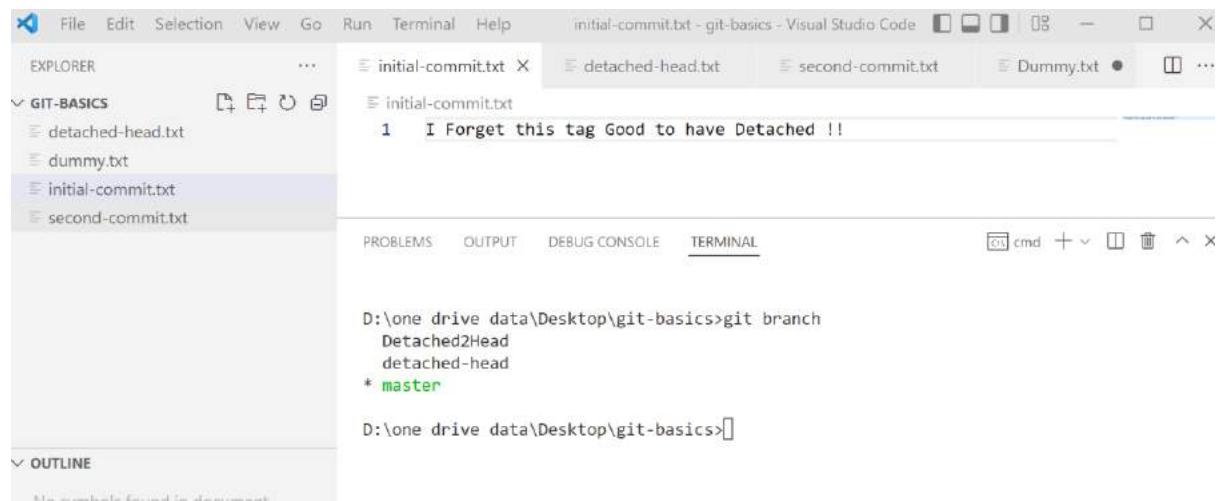
```
Switched to branch 'master'
```

```
D:\one drive data\Desktop\git-basics>git merge Detached2Head
```

```
Merge made by the 'ort' strategy.
```

```
second-commit.txt | 1 +
```

```
1 file changed, 1 insertion(+)
```



```
D:\one drive data\Desktop\git-basics>git branch
Detached2Head
detached-head
* master
```

Fig. Succesful head Changes

```
D:\one drive data\Desktop\git-basics>git branch -D Detached2Head detached-head
```

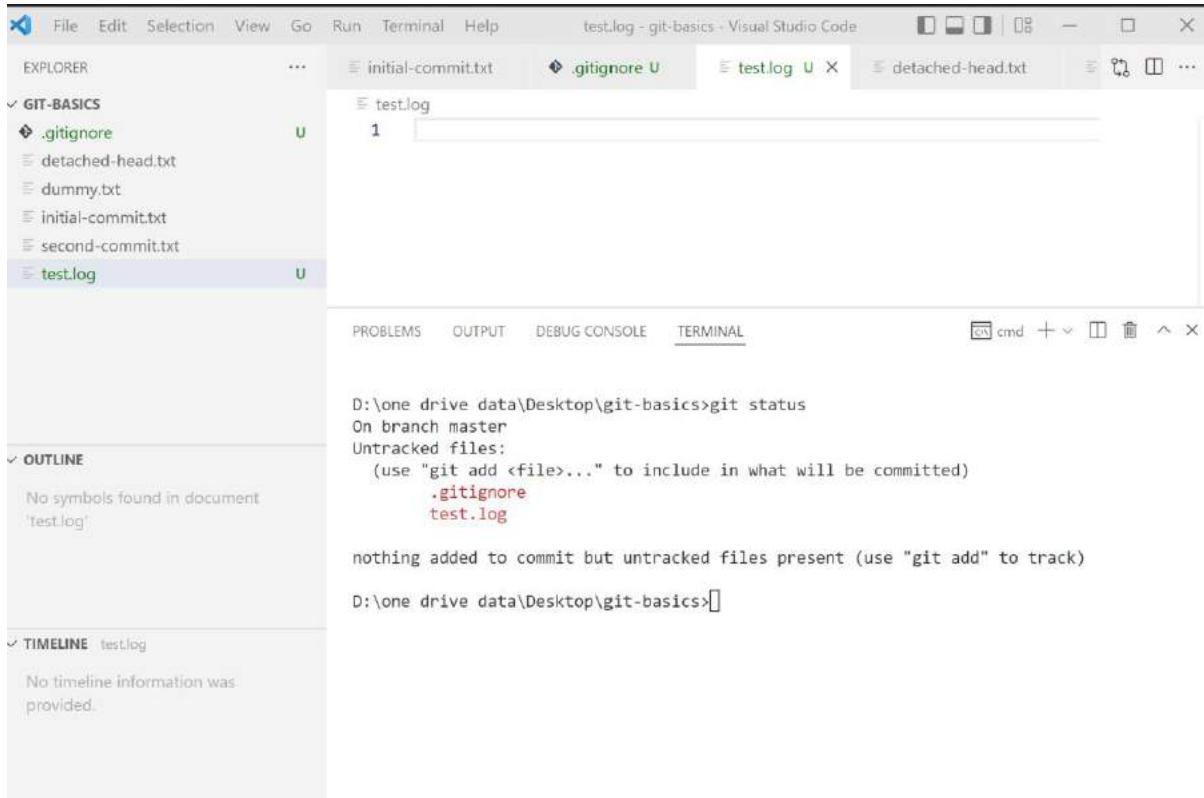
```
Deleted branch Detached2Head (was 4578010).
```

```
Deleted branch detached-head (was b44e221).
```

```
D:\one drive data\Desktop\git-basics>git branch
```

```
* master
```

## Understanding .gitignore:



The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER**: Shows a tree view of files and folders under "GIT-BASICS". The ".gitignore" file is open, containing the entry "test.log".
- TERMINAL**: Displays a command-line session:

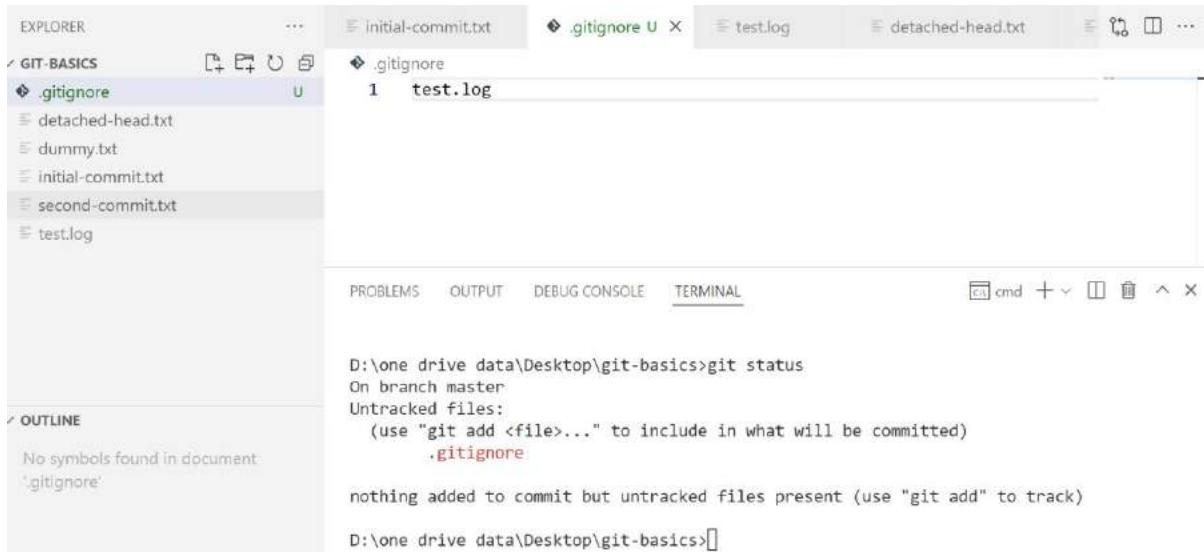
```
D:\one\drive\data\Desktop\git-basics>git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .gitignore
    test.log

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

D:\one\drive\data\Desktop\git-basics>
```

Fig. ignore file created

## Ignoring test.log File



The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER**: Shows a tree view of files and folders under "GIT-BASICS". The ".gitignore" file is open, containing the entry "test.log".
- TERMINAL**: Displays a command-line session:

```
D:\one\drive\data\Desktop\git-basics>git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .gitignore

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

D:\one\drive\data\Desktop\git-basics>
```

Fig. test.log File ignored

## To Ignore Multiple Files:

**\*.log – for ignorance all files**

```
D:\one drive data\Desktop\git-basics>git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .gitignore
    test.log
    test2.log
    test3.log

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

D:\one drive data\Desktop\git-basics>
```

Fig. Multiple File get ignored test, test2 and test3

## Not to Ignore files “!test.log” – not ignoring the files

```
D:\one drive data\Desktop\git-basics>git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .gitignore
    test.log

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

D:\one drive data\Desktop\git-basics>
```

Fig. Not ignored

The screenshot shows the Visual Studio Code interface. In the Explorer sidebar, there is a folder named "GIT-BASICS". Inside "GIT-BASICS", there are several files: "app-data", "index.html", ".gitignore", "detached-head.txt", "dummy.txt", "initial-commit.txt", "second-commit.txt", "test.log", "test2.log", and "test3.log". The ".gitignore" file is selected and open in the editor tab. Its content is:

```
1 *.log
2 !test.log
3 app-data/*
```

The terminal tab shows the command "git status" being run in the directory "D:\one drive data\Desktop\git-basics". The output is:

```
D:\one drive data\Desktop\git-basics>git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .gitignore
    test.log

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

D:\one drive data\Desktop\git-basics>
```

Fig. ignored folders “/” command

The screenshot shows the Visual Studio Code interface. The setup is identical to the first one, with a "GIT-BASICS" folder containing "app-data", "index.html", ".gitignore", and other log files. The ".gitignore" file content is the same as in the first screenshot:

```
1 *.log
2 !test.log
3 app-data/*
```

The terminal tab shows the command "git status" being run. The output is different:

```
D:\one drive data\Desktop\git-basics>git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .gitignore
    app-data/
    test.log

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

D:\one drive data\Desktop\git-basics>
```

Fig. Not ignored folders without “/” command

## Clean all Files & Folders: Deletes untracked Files

D:\one drive data\Desktop\git-basics>git clean -df

The screenshot shows a Visual Studio Code interface. In the Explorer sidebar, there is a folder named 'GIT-BASICS' containing files: detached-head.txt, dummy.txt, initial-commit.txt, and second-commit.txt. In the center, there is a code editor tab for '.gitignore' with the following content:

```
1 *.log
2 !test.log
```

Below the code editor is a terminal window showing the command and its execution:

```
D:\one drive data\Desktop\git-basics>git clean -df
Removing test2.log
Removing test3.log

D:\one drive data\Desktop\git-basics>git ls-files
detached-head.txt
dummy.txt
initial-commit.txt
second-commit.txt

D:\one drive data\Desktop\git-basics>
```

Fig. Clean all files & folders

## Wrap Up What We Learn:

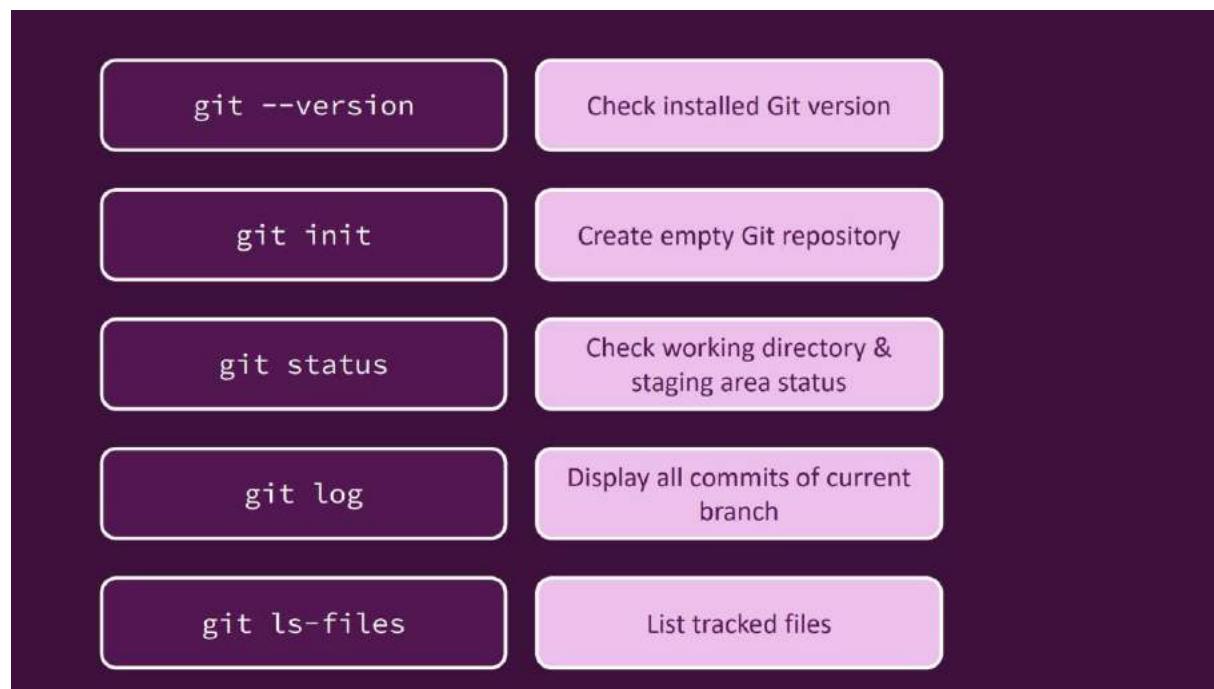


Fig . General Commands

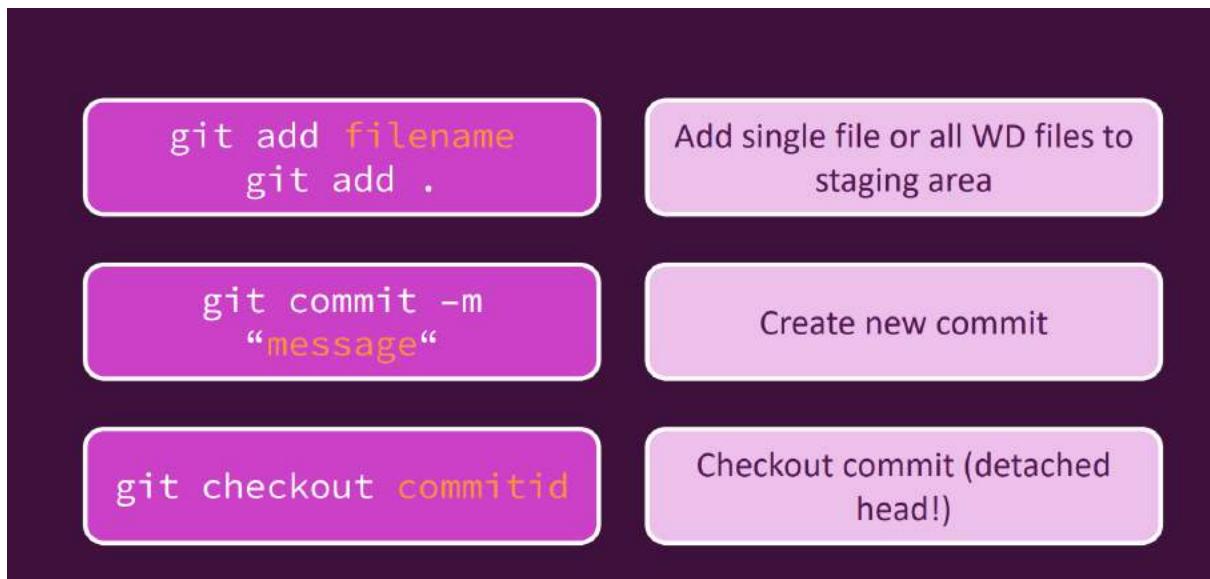


Fig. Basic Commit Creation and access

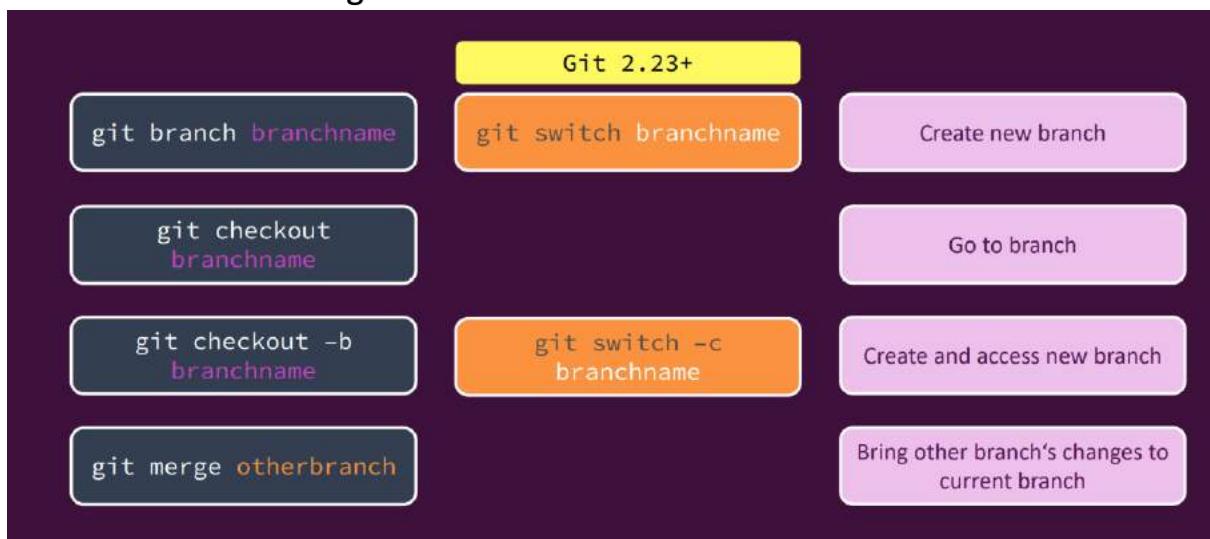


Fig. Branch and creation access



Fig. Deleting Data Summary

# Diving Deep into Git



Fig. Module Introduction

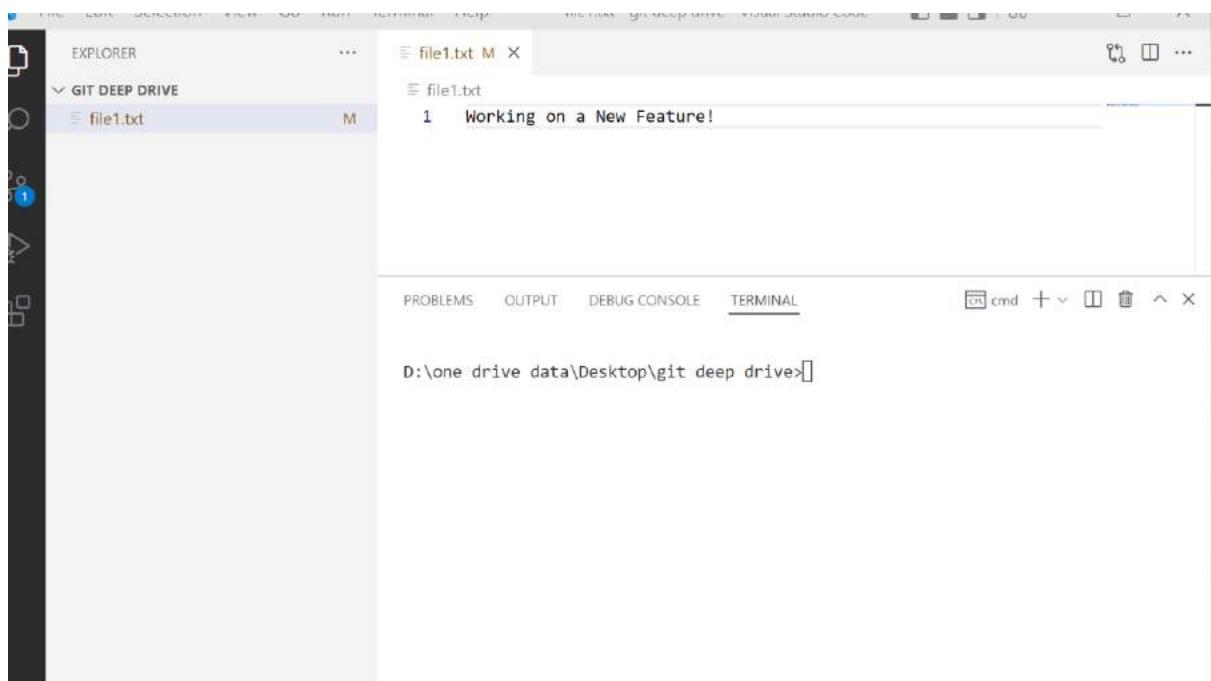
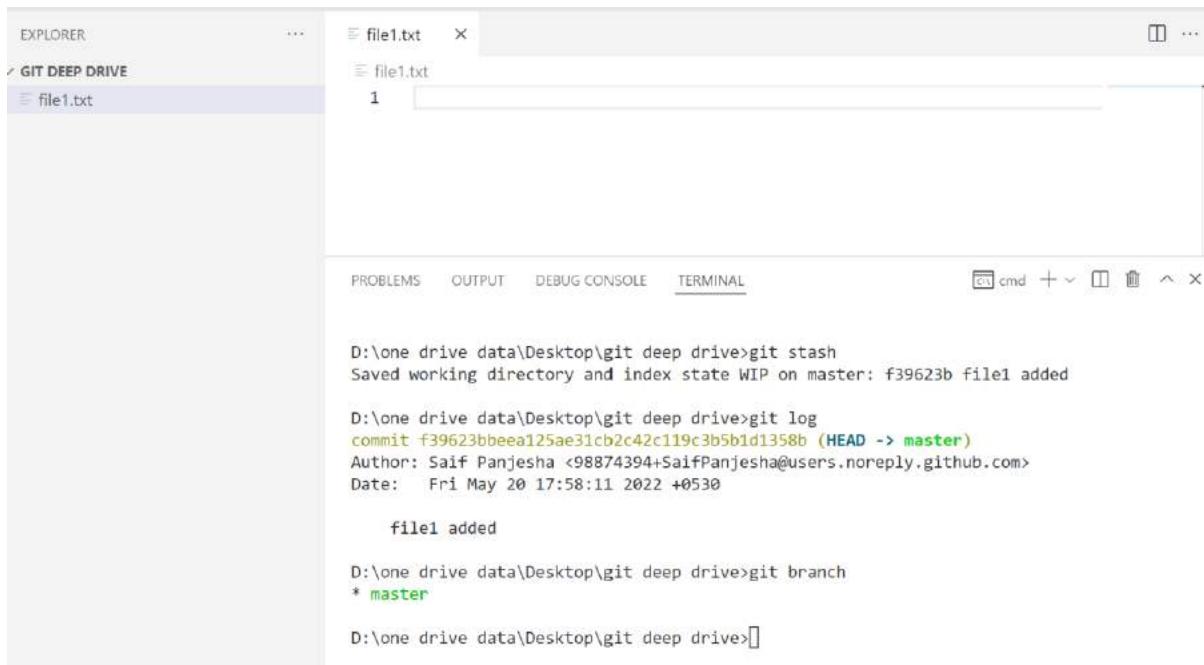


Fig. Created new file into Git Deep Drive Folder

**Understanding the Stash (“git stash”):** The stash is kind of an internal memory where you can save uncommitted Unstaged changes.

**git stash:** temporarily shelves (or stashes) change you've made to your working copy so you can work on something else, and then come back and re-apply them later on.



The screenshot shows the VS Code interface with the terminal tab selected. The terminal window displays the following command and its output:

```
D:\one drive data\Desktop\git deep drive>git stash
Saved working directory and index state WIP on master: f39623b file1 added

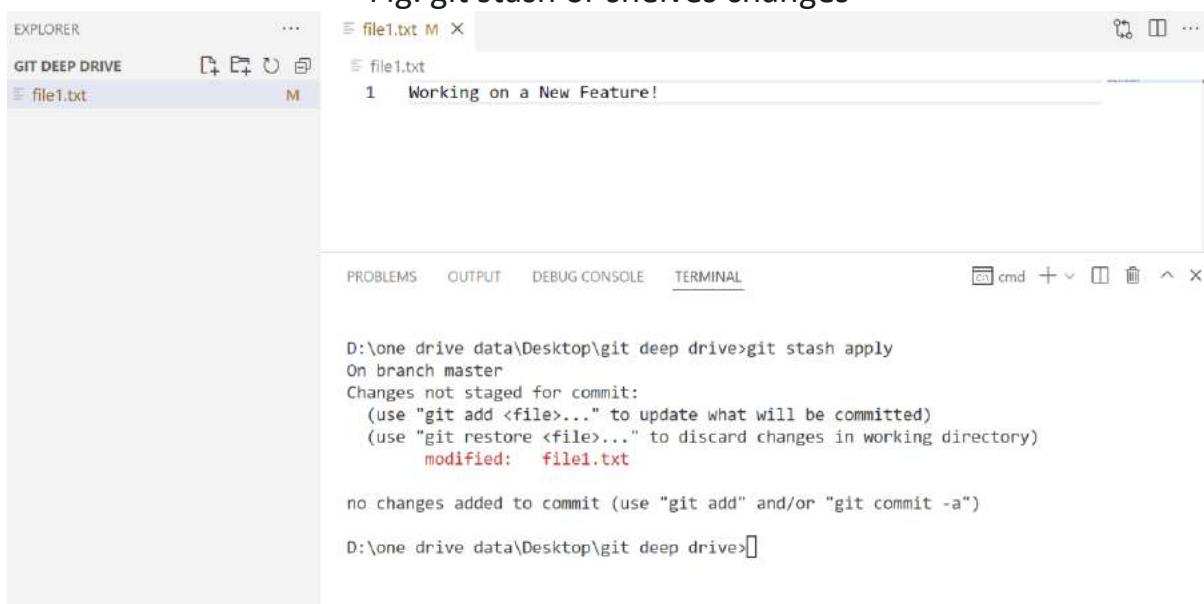
D:\one drive data\Desktop\git deep drive>git log
commit f39623bbeea125ae31cb2c42c119c3b5b1d1358b (HEAD -> master)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 17:58:11 2022 +0530

    file1 added

D:\one drive data\Desktop\git deep drive>git branch
* master

D:\one drive data\Desktop\git deep drive>
```

Fig. git stash or shelves changes



The screenshot shows the VS Code interface with the terminal tab selected. The terminal window displays the following command and its output:

```
D:\one drive data\Desktop\git deep drive>git stash apply
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   file1.txt

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

D:\one drive data\Desktop\git deep drive>
```

Fig. git stash apply

## Continue Working With Another Awesome Feature In Code .



A screenshot of the Visual Studio Code interface. The Explorer sidebar shows a folder named 'GIT DEEP DRIVE' containing a file named 'file1.txt'. The main editor tab is titled 'file1.txt' and contains the following text:

```
1 Working on a New Feature!
2
3 Continue Working With Another Awesome Feature In Code
```

The bottom right corner of the editor has a small preview window showing the first few lines of the file.

The bottom navigation bar includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL. The TERMINAL tab is active, showing the command line prompt 'D:\one drive data\Desktop\git deep drive>'. A small icon for the terminal is also present in the bottom right corner.

Fig. Added New Feature in Code



A screenshot of the Visual Studio Code interface. The Explorer sidebar shows a folder named 'GIT DEEP DRIVE' containing a file named 'file1.txt'. The main editor tab is titled 'file1.txt' and contains the number '1'.

The bottom right corner of the editor has a small preview window showing the first few lines of the file.

The bottom navigation bar includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL. The TERMINAL tab is active, showing the command line output of the 'git stash' command:

```
D:\one drive data\Desktop\git deep drive>git stash
Saved working directory and index state WIP on master: f39623b file1 added
```

A small icon for the terminal is also present in the bottom right corner.

Fig. git stash Work Saved

The screenshot shows the VS Code interface with the Terminal tab selected. In the terminal, the command `git stash apply` is run on a branch named `master`. The output shows that the working directory and index state were saved as WIP on the master branch, and the file `file1.txt` was added. The terminal then prompts for committing changes or discarding them.

```
D:\one drive data\Desktop\git deep drive>git stash
Saved working directory and index state WIP on master: f39623b file1 added

D:\one drive data\Desktop\git deep drive>git stash apply
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   file1.txt

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

D:\one drive data\Desktop\git deep drive>
```

Fig. git stash apply

### git stash list: List of all Stash Changes

The screenshot shows the VS Code interface with the Terminal tab selected. The command `git stash list` is run, displaying three entries in the stash:

- stash@{0}: WIP on master: f39623b file1 added
- stash@{1}: WIP on master: f39623b file1 added
- stash@{2}: WIP on master: f39623b file1 added

```
D:\one drive data\Desktop\git deep drive>git stash list
stash@{0}: WIP on master: f39623b file1 added
stash@{1}: WIP on master: f39623b file1 added
stash@{2}: WIP on master: f39623b file1 added

D:\one drive data\Desktop\git deep drive>
```

Fig. git stash list

The screenshot shows the VS Code interface with the Git Deep Drive extension. In the Explorer sidebar, there is a file named 'file1.txt'. In the terminal tab, the user runs 'git stash list' which shows three stashed changes (stash@{0}, stash@{1}, stash@{2}). Then, they run 'git stash apply 2' which fails because the changes would be overwritten by merge. It suggests committing or stash-ing the changes before merging. The status bar at the bottom indicates the current path is 'D:\one drive data\Desktop\git deep drive'.

```
D:\one drive data\Desktop\git deep drive>git stash list
stash@{0}: WIP on master: f39623b file1 added
stash@{1}: WIP on master: f39623b file1 added
stash@{2}: WIP on master: f39623b file1 added

D:\one drive data\Desktop\git deep drive>git stash apply 2
error: Your local changes to the following files would be overwritten by merge:
  file1.txt
Please commit your changes or stash them before you merge.
Aborting
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   file1.txt

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

D:\one drive data\Desktop\git deep drive>
```

Fig. Error Changes not saved we need to commit, add or stash the changes

The screenshot shows the VS Code interface with the Git Deep Drive extension. In the Explorer sidebar, there is a file named 'file1.txt'. In the terminal tab, the user runs 'git stash' which saves the working directory and index state. Then, they run 'git stash list' which shows four stashed changes (stash@{0} through stash@{3}). The status bar at the bottom indicates the current path is 'D:\one drive data\Desktop\git deep drive'.

```
D:\one drive data\Desktop\git deep drive>git stash
Saved working directory and index state WIP on master: f39623b file1 added

D:\one drive data\Desktop\git deep drive>git stash list
stash@{0}: WIP on master: f39623b file1 added
stash@{1}: WIP on master: f39623b file1 added
stash@{2}: WIP on master: f39623b file1 added
stash@{3}: WIP on master: f39623b file1 added

D:\one drive data\Desktop\git deep drive>
```

Fig. git stash changes saved in working directory

The screenshot shows the VS Code interface with the terminal tab selected. The terminal window displays the following command-line session:

```
D:\one drive data\Desktop\git deep drive>git stash
Saved working directory and index state WIP on master: f39623b file1 added

D:\one drive data\Desktop\git deep drive>git stash list
stash@{0}: WIP on master: f39623b file1 added
stash@{1}: WIP on master: f39623b file1 added
stash@{2}: WIP on master: f39623b file1 added
stash@{3}: WIP on master: f39623b file1 added

D:\one drive data\Desktop\git deep drive>git stash apply 3
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
    (use "git restore <file>..." to discard changes in working directory)
      modified:   file1.txt

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

D:\one drive data\Desktop\git deep drive>[]
```

Fig. git stash apply 2 shows we are working on new feature

## To check our latest changes:

The screenshot shows the VS Code interface with the terminal tab selected. The terminal window displays the following command-line session:

```
D:\one drive data\Desktop\git deep drive>git stash
Saved working directory and index state WIP on master: f39623b file1 added

D:\one drive data\Desktop\git deep drive>git stash list
stash@{0}: WIP on master: f39623b file1 added
stash@{1}: WIP on master: f39623b file1 added
stash@{2}: WIP on master: f39623b file1 added
stash@{3}: WIP on master: f39623b file1 added
stash@{4}: WIP on master: f39623b file1 added

D:\one drive data\Desktop\git deep drive>git stash apply 1
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
    (use "git restore <file>..." to discard changes in working directory)
      modified:   file1.txt

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

Fig. To Check our Latest changes with git stash apply

We can add message to our stash:



Fig. Third Feature added

**stash push -m "Third Feature added"**

```
D:\one drive data\Desktop\git deep drive>git stash push -m "Third Feature added"
Saved working directory and index state On master: Third Feature added

D:\one drive data\Desktop\git deep drive>git stash list
stash@{0}: On master: Third Feature added
stash@{1}: WIP on master: f39623b file1 added
stash@{2}: WIP on master: f39623b file1 added
stash@{3}: WIP on master: f39623b file1 added
stash@{4}: WIP on master: f39623b file1 added
stash@{5}: WIP on master: f39623b file1 added

D:\one drive data\Desktop\git deep drive>
```

The screenshot shows the VS Code interface with the terminal tab active. The terminal window displays the command 'git stash push -m "Third Feature added"' being run, followed by the output which shows the stash entry was saved with the message 'Third Feature added'. Below this, a list of six stash entries is shown, each with a unique identifier, the commit hash, and the file 'file1' added.

Fig. shows we can identify different feature in our stash with the Message

## Now this message we have to add in our Projects:

The screenshot shows a terminal window with the following command history and output:

```
D:\one drive data\Desktop\git deep drive>git stash list
stash@{0}: On master: Third Feature added
stash@{1}: WIP on master: f39623b file1 added
stash@{2}: WIP on master: f39623b file1 added
stash@{3}: WIP on master: f39623b file1 added
stash@{4}: WIP on master: f39623b file1 added
stash@{5}: WIP on master: f39623b file1 added

D:\one drive data\Desktop\git deep drive>git stash pop 0
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   file1.txt

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (a4361775d2171a30ad35e9ac256d6503d4c2c51b)

D:\one drive data\Desktop\git deep drive>git add .
D:\one drive data\Desktop\git deep drive>git commit -m "feture 3 added to project"
[master 7faede4] feture 3 added to project
 1 file changed, 5 insertions(+)

D:\one drive data\Desktop\git deep drive>git stash list
stash@{0}: WIP on master: f39623b file1 added
stash@{1}: WIP on master: f39623b file1 added
stash@{2}: WIP on master: f39623b file1 added
stash@{3}: WIP on master: f39623b file1 added
stash@{4}: WIP on master: f39623b file1 added

D:\one drive data\Desktop\git deep drive>
```

```
D:\one drive data\Desktop\git deep drive>git log
commit 7faede4ff4b3e58ccb6580c10094065c7458367b (HEAD -> master)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 18:33:36 2022 +0530

  feture 3 added to project

commit f39623bbeea125ae31cb2c42c119c3b5b1d1358b
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 17:58:11 2022 +0530

  file1 added

D:\one drive data\Desktop\git deep drive>
```

Fig. Added Successful to project and out from stash stack

## git stash clear: Remove the git stash

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    cmd +

```
D:\one drive data\Desktop\git deep drive>git stash list
stash@{0}: WIP on master: f39623b file1 added
stash@{1}: WIP on master: f39623b file1 added
stash@{2}: WIP on master: f39623b file1 added
stash@{3}: WIP on master: f39623b file1 added
stash@{4}: WIP on master: f39623b file1 added
```

```
D:\one drive data\Desktop\git deep drive>git stash drop 0
Dropped refs/stash@{0} (bd31c4c70ae42847ceb8a65e20970a22b8f53614)
```

```
D:\one drive data\Desktop\git deep drive>git stash list
stash@{0}: WIP on master: f39623b file1 added
stash@{1}: WIP on master: f39623b file1 added
stash@{2}: WIP on master: f39623b file1 added
stash@{3}: WIP on master: f39623b file1 added
```

```
D:\one drive data\Desktop\git deep drive>git stash clear
```

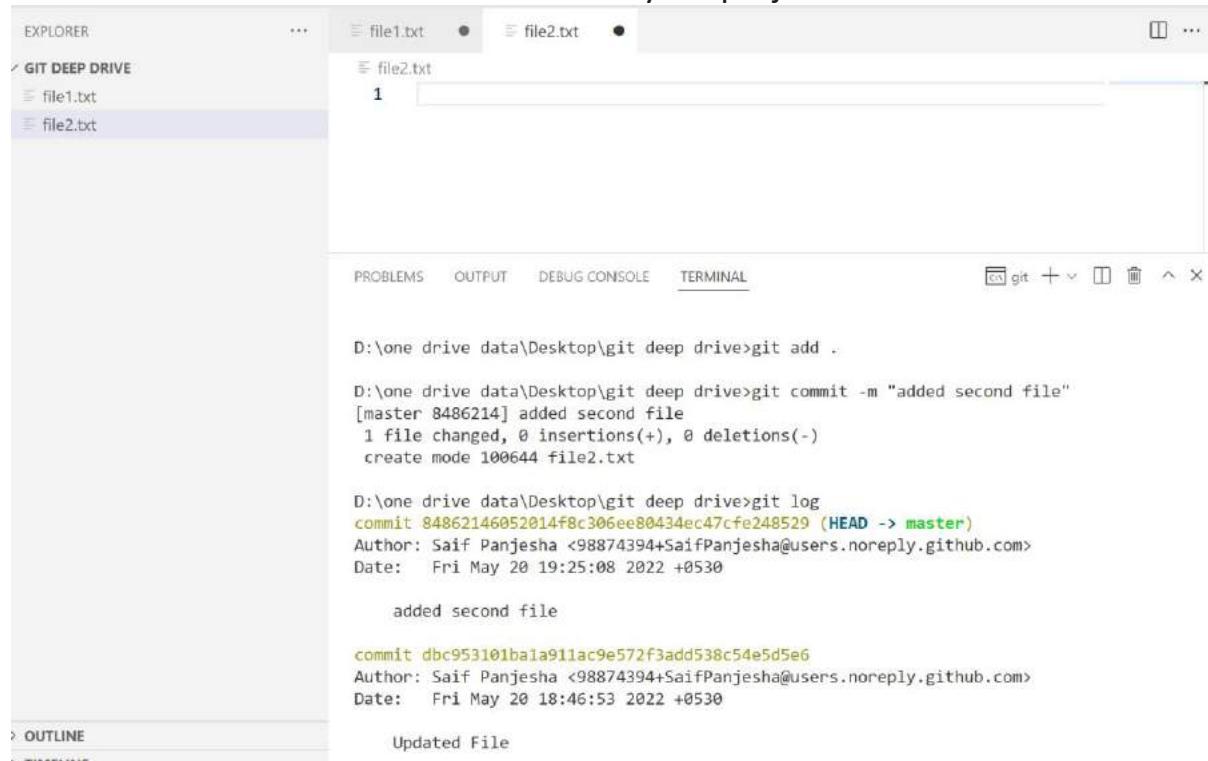
```
D:\one drive data\Desktop\git deep drive>git stash list
```

```
D:\one drive data\Desktop\git deep drive>[]
```

Fig. Remove the stash

## Bringing the Lost data with “git reflog”

Let's create a new file file2.txt and add in your project



The screenshot shows the VS Code interface with the Explorer sidebar open, displaying a folder named "GIT DEEP DRIVE" containing "file1.txt" and "file2.txt". The "file2.txt" tab is active, showing the number "1" in the editor area. Below the editor are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL. The TERMINAL tab is selected, showing the command history:

```
D:\one drive data\Desktop\git deep drive>git add .
D:\one drive data\Desktop\git deep drive>git commit -m "added second file"
[master 8486214] added second file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file2.txt

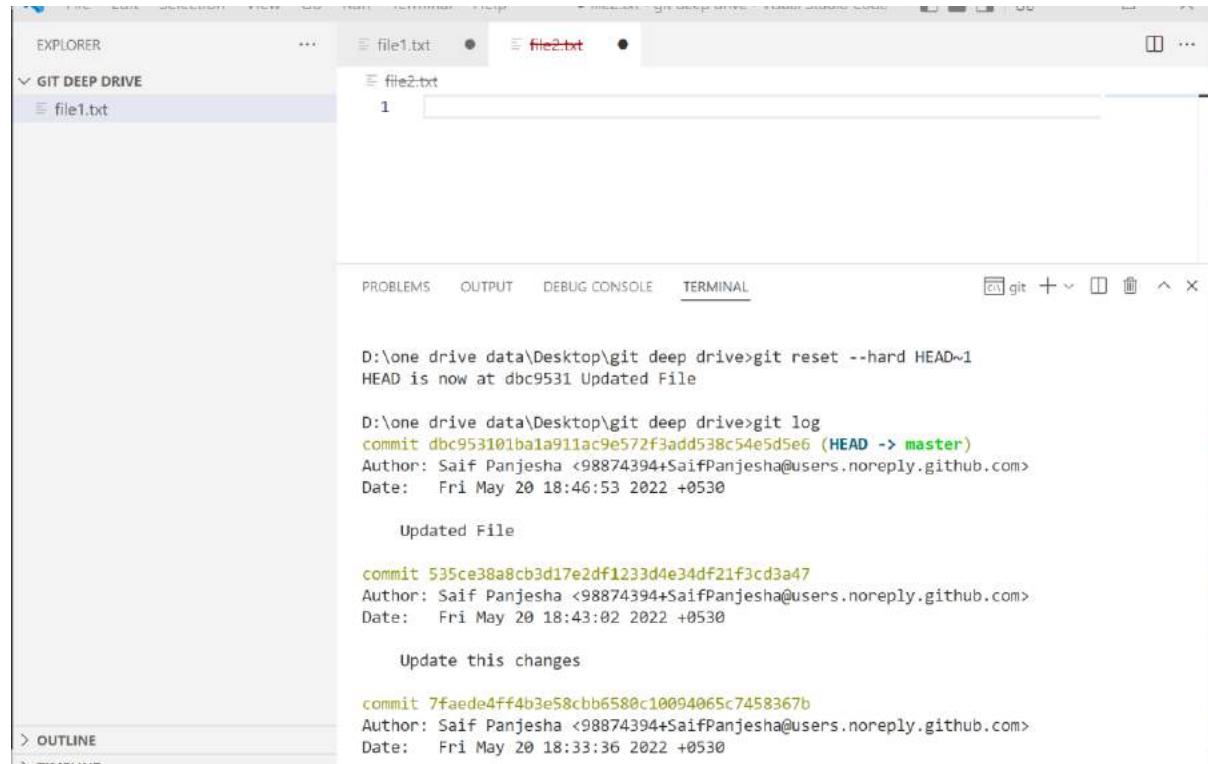
D:\one drive data\Desktop\git deep drive>git log
commit 84862146052014f8c306ee80434ec47cfe248529 (HEAD -> master)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 19:25:08 2022 +0530

    added second file

commit dbc953101b1a911ac9e572f3add538c54e5d5e6
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 18:46:53 2022 +0530

Updated File
```

Fig. file2.txt created



The screenshot shows the VS Code interface with the Explorer sidebar open, displaying a folder named "GIT DEEP DRIVE" containing "file1.txt". The "file2.txt" tab is active, showing the number "1" in the editor area. Below the editor are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL. The TERMINAL tab is selected, showing the command history:

```
D:\one drive data\Desktop\git deep drive>git reset --hard HEAD~1
HEAD is now at dbc9531 Updated File

D:\one drive data\Desktop\git deep drive>git log
commit dbc953101b1a911ac9e572f3add538c54e5d5e6 (HEAD -> master)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 18:46:53 2022 +0530

    Updated File

commit 535ce38a8cb3d17e2df1233d4e34df21f3cd3a47
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 18:43:02 2022 +0530

    Update this changes

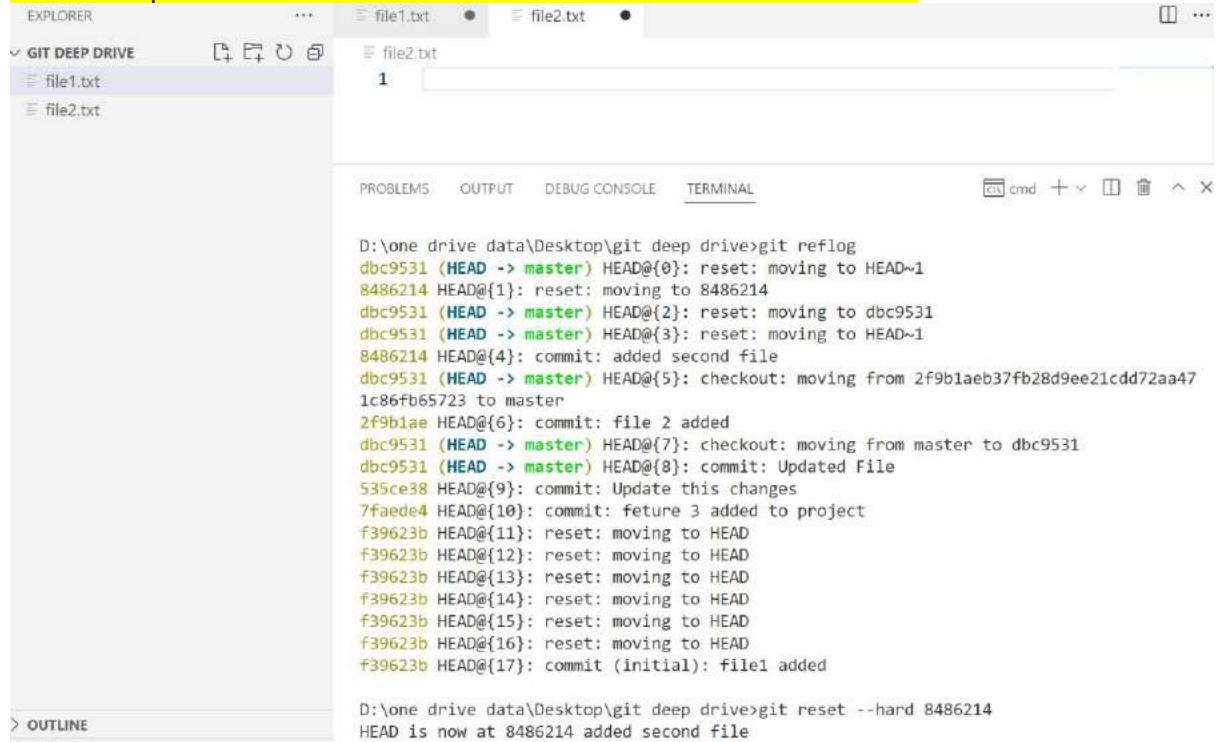
commit 7faede4ff4b3e8cbb6580c10094065c7458367b
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 18:33:36 2022 +0530
```

Fig. removed file not in used

Now to restore for next operations the file we need “git reflog”:

Git keeps track of updates to the tip of branches using a mechanism called reference logs, or "reflogs."

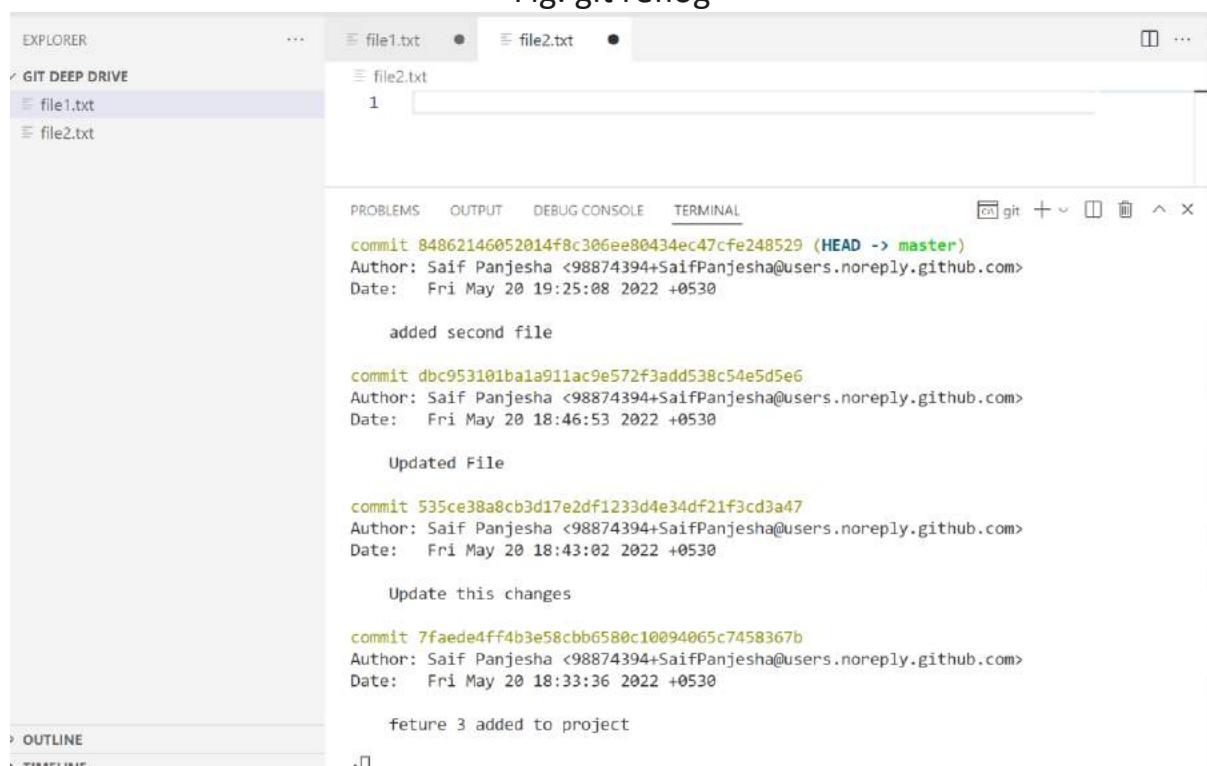
A branch tip is the last commit or most recent commit on a branch



```
D:\one drive data\Desktop\git deep drive>git reflog
dbc9531 (HEAD -> master) HEAD@{0}: reset: moving to HEAD~1
8486214 HEAD@{1}: reset: moving to 8486214
dbc9531 (HEAD -> master) HEAD@{2}: reset: moving to dbc9531
dbc9531 (HEAD -> master) HEAD@{3}: reset: moving to HEAD~1
8486214 HEAD@{4}: commit: added second file
dbc9531 (HEAD -> master) HEAD@{5}: checkout: moving from 2f9b1aeb37fb28d9ee21cdd72aa47
1c86fb65723 to master
2f9b1ae HEAD@{6}: commit: file 2 added
dbc9531 (HEAD -> master) HEAD@{7}: checkout: moving from master to dbc9531
dbc9531 (HEAD -> master) HEAD@{8}: commit: Updated File
535ce38 HEAD@{9}: commit: Update this changes
7faede4 HEAD@{10}: commit: fature 3 added to project
f39623b HEAD@{11}: reset: moving to HEAD
f39623b HEAD@{12}: reset: moving to HEAD
f39623b HEAD@{13}: reset: moving to HEAD
f39623b HEAD@{14}: reset: moving to HEAD
f39623b HEAD@{15}: reset: moving to HEAD
f39623b HEAD@{16}: reset: moving to HEAD
f39623b HEAD@{17}: commit (initial): file1 added

D:\one drive data\Desktop\git deep drive>git reset --hard 8486214
HEAD is now at 8486214 added second file
```

Fig. git reflog



```
commit 84862146052014f8c306ee80434ec47cf248529 (HEAD -> master)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 19:25:08 2022 +0530

    added second file

commit dbc953101ba1a911ac9e572f3add538c54e5d5e6
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 18:46:53 2022 +0530

    Updated File

commit 535ce38a8cb3d17e2df1233d4e34df21f3cd3a47
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 18:43:02 2022 +0530

    Update this changes

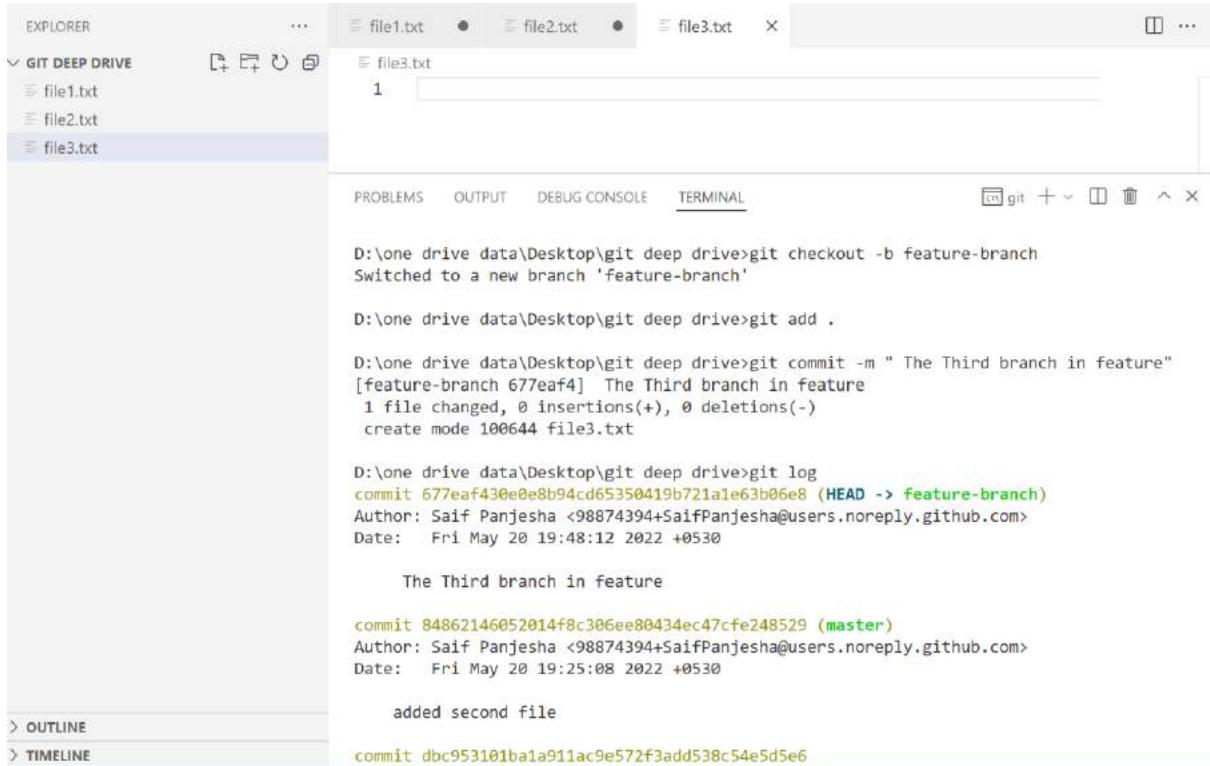
commit 7faede4ff4b3e58cbb6580c10094065c7458367b
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 18:33:36 2022 +0530

    fature 3 added to project
```

Fig. Successful added second file with commit history

## “reflog” in Branches:

Let's create a new branch called **feature\_branch** and add file3.txt file



The screenshot shows the VS Code interface with the following details:

- EXPLORER** panel: Shows files file1.txt, file2.txt, and file3.txt.
- TERMINAL** tab: Displays the command history for creating a new branch and committing changes.
- OUTPUT** tab: Shows the log output of the git commands.
- DEBUG CONSOLE**: Not used.
- PROBLEMS**: Not used.
- git** icon in the terminal header.

```
D:\one\drive\data\Desktop\git\deep\drive>git checkout -b feature-branch
Switched to a new branch 'feature-branch'

D:\one\drive\data\Desktop\git\deep\drive>git add .
D:\one\drive\data\Desktop\git\deep\drive>git commit -m "The Third branch in feature"
[feature-branch 677eaf4] The Third branch in feature
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file3.txt

D:\one\drive\data\Desktop\git\deep\drive>git log
commit 677eaf430e0e8b94cd65350419b721a1e63b06e8 (HEAD -> feature-branch)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 19:48:12 2022 +0530

  The Third branch in feature

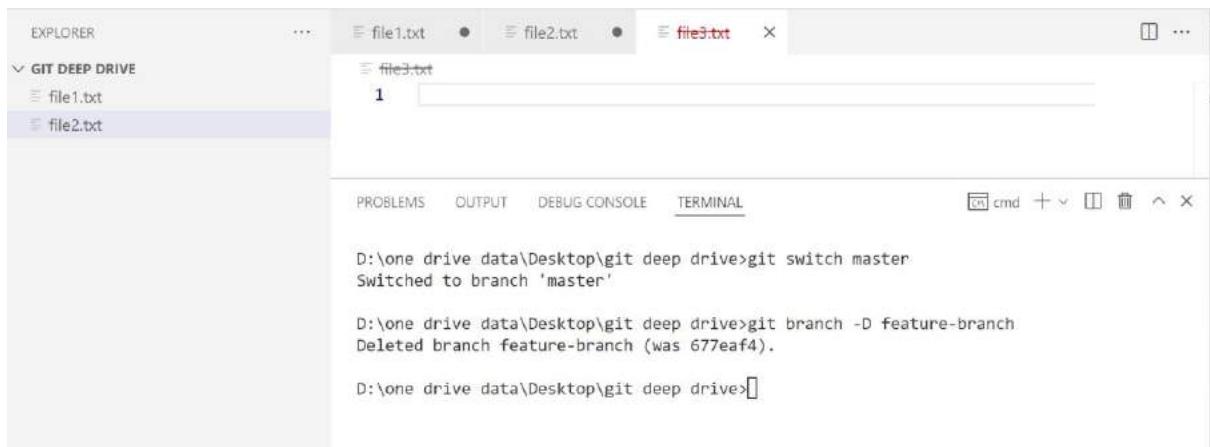
commit 84862146052014f8c306ee80434ec47cf248529 (master)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 19:25:08 2022 +0530

  added second file

commit dbc953101ba1a911ac9e572f3add538c54e5d5e6
```

Fig. created a new branch with file3.txt

## Switch back to master branch:



The screenshot shows the VS Code interface with the following details:

- EXPLORER** panel: Shows files file1.txt, file2.txt, and file3.txt.
- TERMINAL** tab: Displays the command history for switching branches and deleting the feature branch.
- OUTPUT** tab: Shows the log output of the git commands.
- DEBUG CONSOLE**: Not used.
- PROBLEMS**: Not used.
- cmd** icon in the terminal header.

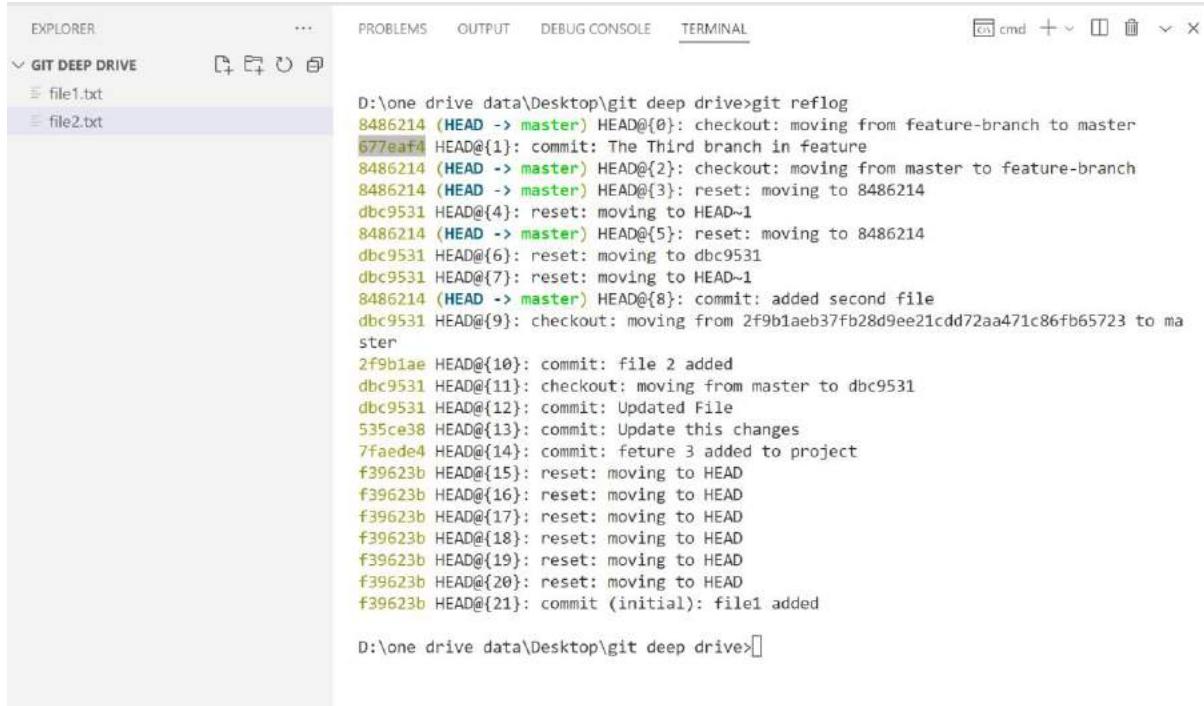
```
D:\one\drive\data\Desktop\git\deep\drive>git switch master
Switched to branch 'master'

D:\one\drive\data\Desktop\git\deep\drive>git branch -D feature-branch
Deleted branch feature-branch (was 677eaf4).

D:\one\drive\data\Desktop\git\deep\drive>
```

Fig. feature and file3.txt deleted

For restore the “**branches**” is differs from “**commit**”



D:\one drive data\Desktop\git deep drive>git reflog  
8486214 (HEAD -> master) HEAD@{0}: checkout: moving from feature-branch to master  
677eaf4 HEAD@{1}: commit: The Third branch in feature  
8486214 (HEAD -> master) HEAD@{2}: checkout: moving from master to feature-branch  
8486214 (HEAD -> master) HEAD@{3}: reset: moving to 8486214  
dbc9531 HEAD@{4}: reset: moving to HEAD~1  
8486214 (HEAD -> master) HEAD@{5}: reset: moving to 8486214  
dbc9531 HEAD@{6}: reset: moving to dbc9531  
dbc9531 HEAD@{7}: reset: moving to HEAD~1  
8486214 (HEAD -> master) HEAD@{8}: commit: added second file  
dbc9531 HEAD@{9}: checkout: moving from 2f9b1aeb37fb28d9ee21cdd72aa471c86fb65723 to master  
2f9b1ae HEAD@{10}: commit: file 2 added  
dbc9531 HEAD@{11}: checkout: moving from master to dbc9531  
dbc9531 HEAD@{12}: commit: Updated File  
535ce38 HEAD@{13}: commit: Update this changes  
7faede4 HEAD@{14}: commit: future 3 added to project  
f39623b HEAD@{15}: reset: moving to HEAD  
f39623b HEAD@{16}: reset: moving to HEAD  
f39623b HEAD@{17}: reset: moving to HEAD  
f39623b HEAD@{18}: reset: moving to HEAD  
f39623b HEAD@{19}: reset: moving to HEAD  
f39623b HEAD@{20}: reset: moving to HEAD  
f39623b HEAD@{21}: commit (initial): file1 added  
  
D:\one drive data\Desktop\git deep drive>

Fig. git reflog help us to get **Git keeps track of updates to the tip of branches**



D:\one drive data\Desktop\git deep drive>git checkout 677eaf4  
Note: switching to '677eaf4'.  
  
You are in 'detached HEAD' state. You can look around, make experimental changes and commit them, and you can discard any commits you make in this state without impacting any branches by switching back to a branch.  
  
If you want to create a new branch to retain commits you create, you may do so (now or later) by using -c with the switch command. Example:  
  
git switch -c <new-branch-name>  
  
Or undo this operation with:  
  
git switch -  
  
Turn off this advice by setting config variable advice.detachedHead to false  
  
HEAD is now at 677eaf4 The Third branch in feature  
  
D:\one drive data\Desktop\git deep drive>

Fig. updated track ID has added to detached head

The screenshot shows the VS Code interface with the following details:

- Explorer View:** Shows a tree view with a expanded folder named "GIT DEEP DRIVE". Inside, there are three files: "file1.txt", "file2.txt", and "file3.txt".
- Terminal Tab:** Active tab, showing the command line interface.
- Terminal Output:**
  - Command: `D:\one drive data\Desktop\git deep drive>git checkout 677eaf4`
  - Output: "Note: switching to '677eaf4'."
  - Message: "You are in 'detached HEAD' state. You can look around, make experimental changes and commit them, and you can discard any commits you make in this state without impacting any branches by switching back to a branch."
  - Message: "If you want to create a new branch to retain commits you create, you may do so (now or later) by using -c with the switch command. Example:
  - Text: `git switch -c <new-branch-name>`
  - Text: "Or undo this operation with:
  - Text: `git switch -`
  - Text: "Turn off this advice by setting config variable advice.detachedHead to false"
  - Text: "HEAD is now at 677eaf4 The Third branch in feature"
  - Code Block:** A highlighted code block in the terminal window:

```
D:\one drive data\Desktop\git deep drive>git switch -c feature_branch
Switched to a new branch 'feature_branch'
```
  - Code Block:** Another highlighted code block in the terminal window:

```
D:\one drive data\Desktop\git deep drive>git branch
* feature_branch
  master
```
  - Text: `D:\one drive data\Desktop\git deep drive>`
- Bottom Navigation:** Shows "OUTLINE" and other standard VS Code navigation icons.

Fig. successful added branches with “reflog”

EXPLORER    ...    PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    git + ⌂ ⌂ ⌂ ⌂

✓ GIT DEEP DRIVE   

D:\one drive data\Desktop\git deep drive>git log

commit 677eaf430e0e8b94cd65350419b721a1e63b06e8 (**HEAD -> feature\_branch**)  
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>  
Date: Fri May 20 19:48:12 2022 +0530

The Third branch in feature

commit 84862146052014f8c306ee80434ec47cf248529 (**master**)  
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>  
Date: Fri May 20 19:25:08 2022 +0530

    added second file

commit dbc953101ba1a911ac9e572f3add538c54e5d5e6  
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>  
Date: Fri May 20 18:46:53 2022 +0530

    Updated File

commit 535ce38a8cb3d17e2df1233d4e34df21f3cd3a47  
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>  
Date: Fri May 20 18:43:02 2022 +0530

    Update this changes

commit 7faede4ff4b3e58ccb6580c10094065c7458367b  
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>  
Date: Fri May 20 18:33:36 2022 +0530

    feature 3 added to project

> OUTLINE

> TIMELINE

Fig. Commit history of current branch(feature\_branch)

## Combining Branches – What & Why?

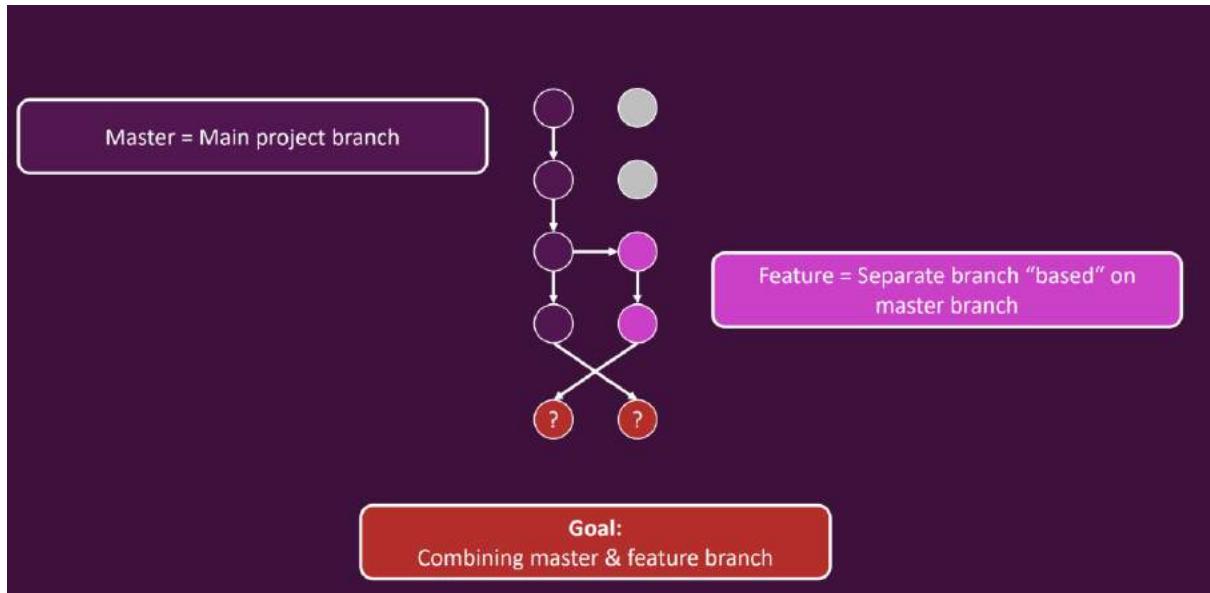


Fig. show Main Branch as Master and Separate Branch as feature

## Understanding Merge Types:



Fig. Shows Types of Merges in Git

## Applying the Fast-Forward Merge:

Fast forward merge can be performed when there is a direct linear path from the source branch to the target branch. In fast-forward merge, git simply moves the source branch pointer to the target branch pointer without creating an extra merge commit.

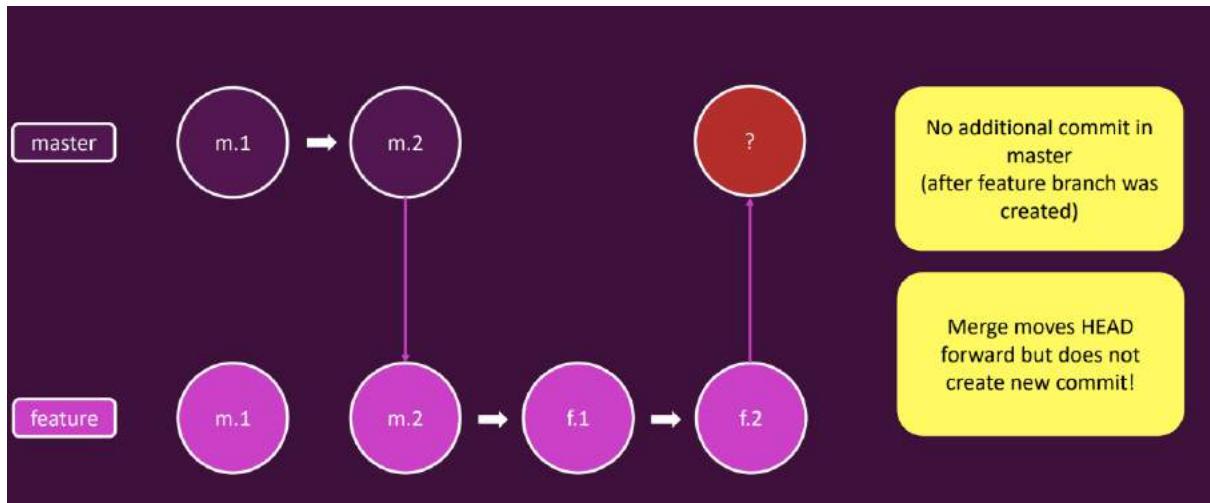


Fig. Shows Master & Feature -Merge("Fast-Forward")

Let's go and create first master branch with two files m1.txt first commit

```
D:\one drive data\Desktop\Branches>git init
Initialized empty Git repository in D:/one drive data/Desktop/Branches/.git/
D:\one drive data\Desktop\Branches>git add .
D:\one drive data\Desktop\Branches>git commit -m "m1 added"
[master (root-commit) 262d300] m1 added
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 master/m1.txt
D:\one drive data\Desktop\Branches>git log
commit 262d300f2eaed871e02d14db2de741302f25bdb4 (HEAD -> master)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:07:28 2022 +0530
    m1 added
D:\one drive data\Desktop\Branches>
```

Fig. Created master branch with first commit

Let's go and create first master branch with two files m2.txt second commit

The screenshot shows the VS Code interface with the following details:

- EXPLORER**: Shows a tree view with a **BRANCHES** section containing a **master** branch which has **m1.txt** and **m2.txt**.
- TERMINAL**: Displays the command-line history:
  - D:\one drive data\Desktop\Branches>git add .
  - D:\one drive data\Desktop\Branches>git commit -m "m2 added"
  - [master bddb6aa] m2 added
  - 1 file changed, 0 insertions(+), 0 deletions(-)
  - create mode 100644 master/m2.txt
  - D:\one drive data\Desktop\Branches>git log
  - commit bddb6aa98ae81dee33d489dd5cedab0a91aec0c2 (HEAD -> master)
  - Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
  - Date: Fri May 20 23:10:20 2022 +0530
  - m2 added
  - commit 262d300f2eaed871e02d14db2de741302f25bdb4
  - Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
  - Date: Fri May 20 23:07:28 2022 +0530
  - m1 added

Fig. Created master branch with second commit

Now, lets create a new branch called “Feature” branch with two files f1.txt first commit and f2.txt second commit

The screenshot shows the VS Code interface with the following details:

- EXPLORER**: Shows a tree view with a **OPEN EDITORS** section containing a **feature** branch which has **f1.txt** and **f2.txt**, and a **master** branch.
- TERMINAL**: Displays the command-line history:
  - D:\one drive data\Desktop\Branches>git switch -c feature
  - Switched to a new branch 'feature'
  - D:\one drive data\Desktop\Branches>git add .
  - D:\one drive data\Desktop\Branches>git commit -m "f1 added"
  - [feature b6fc683] f1 added
  - 1 file changed, 0 insertions(+), 0 deletions(-)
  - create mode 100644 feature/f1.txt
  - D:\one drive data\Desktop\Branches>git add .
  - D:\one drive data\Desktop\Branches>git commit -m "f2 added"
  - [feature 89d84fc] f2 added
  - 1 file changed, 0 insertions(+), 0 deletions(-)
  - create mode 100644 feature/f2.txt
  - D:\one drive data\Desktop\Branches>git log
  - commit B9d84fcraf79eba8ce29224f1cf141fba98cef34 (HEAD -> feature)
  - Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
  - Date: Fri May 20 23:22:19 2022 +0530
  - f2 added'
  - commit b6fc683a2918b3f8b1da9198426b888192c4df43
  - Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
  - Date: Fri May 20 23:21:42 2022 +0530
  - f1 added

Fig. Created feature branch with two files f1.txt and f2.txt

The screenshot shows the Visual Studio Code interface with the terminal tab selected. The terminal window displays the following command-line session:

```
D:\one drive data\Desktop\Branches>git branch
* feature
  master

D:\one drive data\Desktop\Branches>git log
commit 89d84fcacf79eba8ce29224f1cf141fba98cefa34 (HEAD -> feature)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:22:19 2022 +0530

    f2 added'

commit b6fc683a2918b3f8b1da9198426b888192c4df43
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:21:42 2022 +0530

    f1 added

commit f32b9469d954ae7e4b2082b9f14435494697f390 (master)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:17:14 2022 +0530

    m2 added

commit 09ac428c3a0b18850aec1c2f611898d0e42ab706
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:16:24 2022 +0530

    m1 added
```

Fig. Created feature branch with two files and checking commit history

The screenshot shows the Visual Studio Code interface with the terminal tab selected. The terminal window displays the following command-line session:

```
D:\one drive data\Desktop\Branches>git switch master
Switched to branch 'master'

D:\one drive data\Desktop\Branches>
```

Fig. Switched to master branch

## Fast forward merge happens:

D:\one drive data\Desktop\Branches>git switch master  
Switched to branch 'master'  
D:\one drive data\Desktop\Branches>git merge feature  
Updating f32b946..89d84fc  
Fast-forward  
 feature/f1.txt | 0  
 feature/f2.txt | 0  
 2 files changed, 0 insertions(+), 0 deletions(-)  
 create mode 100644 feature/f1.txt  
 create mode 100644 feature/f2.txt  
D:\one drive data\Desktop\Branches>

Fig. shows merged with master branch as fast - forward

D:\one drive data\Desktop\Branches>git log  
commit 89d84fcraf79eba8ce29224f1cf141fba98cefa34 (HEAD -> master, feature)  
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>  
Date: Fri May 20 23:22:19 2022 +0530  
  
 f2 added  
  
commit b6fc683a2918b3f8b1da9198426b888192c4df43  
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>  
Date: Fri May 20 23:21:42 2022 +0530  
  
 f1 added  
  
commit f32b9469d954ae7e4b2082b9f14435494697f390  
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>  
Date: Fri May 20 23:17:14 2022 +0530  
  
 m2 added  
  
commit 09ac428c3a0b18850aec1c2f611898d0e42ab706  
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>  
Date: Fri May 20 23:16:24 2022 +0530  
  
 m1 added

Fig. direct linear path from the source branch to the target branch

```
D:\one\drive\data\Desktop\Branches>git reset --hard HEAD~2
HEAD is now at f32b946 m2 added

D:\one\drive\data\Desktop\Branches>git log
commit f32b9469d954ae7e4b2082b9f14435494697f390 (HEAD -> master)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:17:14 2022 +0530

    m2 added

commit 09ac428c3a0b18850aec1c2f611898d0e42ab706
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:16:24 2022 +0530

    m1 added

D:\one\drive\data\Desktop\Branches>
```

Fig. reset the head to remove extra commit

Note:

Squash will simply well, kind of squash or put together all the commits we had in our feature branch into the latest commit so to say. So only one commit is added to our master branch in the end. Therefore, if we now use git merge --squash feature, you can see that we still have a Fast-forward merge here but if we now check our Git log, you see well, we don't have any commit, right? Now, what's wrong here? Well, nothing is wrong. If you quickly scroll up a bit, you see that the head was not updated here because as I said, with the squash command, with the squash flag, we will put together all the changes made in the feature branch into one single commit and this commit is not our f2 commit here, it is, in the end, as it contains all the changes, but we have to create a separate commit

The screenshot shows the VS Code interface with the terminal tab selected. The terminal window displays the following command-line session:

```
D:\one\drive\data\Desktop\Branches>git branch
  feature
* master

D:\one\drive\data\Desktop\Branches>git merge --squash feature
Updating f32b946..76e4b63
Fast-forward
  Squash commit -- not updating HEAD
    feature/f1.txt | 0
    feature/f2.txt | 0
  2 files changed, 0 insertions(+), 0 deletions(-)
  create mode 100644 feature/f1.txt
  create mode 100644 feature/f2.txt

D:\one\drive\data\Desktop\Branches>git log
commit f32b9469d954ae7e4b2082b9f14435494697f390 (HEAD -> master)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:17:14 2022 +0530

    m2 added

commit 09ac428c3a0b18850aec1c2f611898d0e42ab706
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:16:24 2022 +0530

    m1 added

D:\one\drive\data\Desktop\Branches>
```

The screenshot shows the VS Code interface with the terminal tab selected. The terminal window displays the following command-line session:

```
D:\one\drive\data\Desktop\Branches>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:  feature/f1.txt
    new file:  feature/f2.txt

D:\one\drive\data\Desktop\Branches>git commit -m "together master and feature"
[master 00becf0] together master and feature
  2 files changed, 0 insertions(+), 0 deletions(-)
  create mode 100644 feature/f1.txt
  create mode 100644 feature/f2.txt

D:\one\drive\data\Desktop\Branches>git log
commit 00becf01b9225222dd53b7735859e92eb4746c31 (HEAD -> master)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Sat May 21 00:09:37 2022 +0530

  together master and feature

commit f32b9469d954ae7e4b2082b9f14435494697f390
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:17:14 2022 +0530

    m2 added

commit 09ac428c3a0b18850aec1c2f611898d0e42ab706
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:16:24 2022 +0530
```

Fig. squash the previous commits into one.

## Cleaning the directory:

The screenshot shows a terminal window with the following content:

```
D:\one\drive\data\Desktop\Branches>git log
commit 00becf01b9225222dd53b7735859e92eb4746c31 (HEAD -> master)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Sat May 21 00:09:37 2022 +0530

    together master and feature

commit f32b9469d954ae7e4b2082b9f14435494697f390
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:17:14 2022 +0530

    m2 added

commit 09ac428c3a0b18850aec1c2f611898d0e42ab706
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:16:24 2022 +0530

    m1 added

D:\one\drive\data\Desktop\Branches>git reset --hard HEAD~1
HEAD is now at f32b946 m2 added

D:\one\drive\data\Desktop\Branches>git status
On branch master
nothing to commit, working tree clean

D:\one\drive\data\Desktop\Branches>
```

Fig. Cleaning the directory.

## The Recursive Merge (Non – Fast Forward):

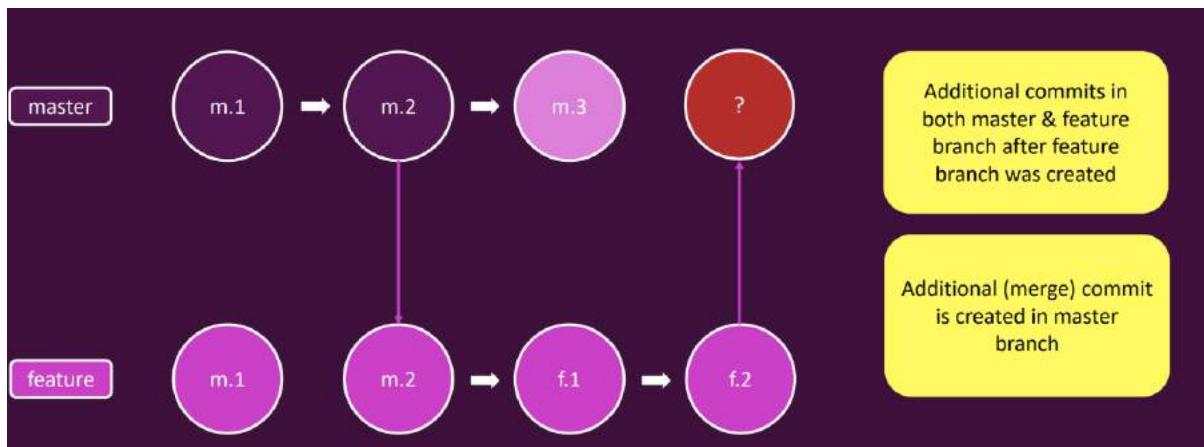


Fig. Shows Master & Feature – Recursive Merge("Non -Fast-Forward")

```

EXPLORER          ...   m1.txt    m2.txt    f1.txt    f2.txt    ...
OPEN EDITORS      ...   feature > f2.txt
BRANCHES         D+ ⌂ ⌂ ⌂
feature
master
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL
D:\one drive data\Desktop\Branches>git merge --no-ff feature
Merge made by the 'ort' strategy.
 feature/f1.txt | 0
 feature/f2.txt | 0
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 feature/f1.txt
 create mode 100644 feature/f2.txt
D:\one drive data\Desktop\Branches>

```

Fig. recursive merge has been added **git merge- - no-ff feature**

```

EXPLORER          ...   PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL
OPEN EDITORS      ...   cmd + ⌂ ⌂ ⌂
BRANCHES         D+ ⌂ ⌂ ⌂
feature
master
D:\one drive data\Desktop\Branches>git log
commit 3bc4230bd61dbb4eb27606a474ac63c232feb909 (HEAD -> master)
Merge: f32b946 76e4b63
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Sat May 21 00:44:43 2022 +0530

Merge branch 'feature'

commit 76e4b63e5cb83abb78c0f5e1861f2e1c11b4754a (feature)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:51:35 2022 +0530

f2 added

commit fb7ff2718002060f6cd541f9c635b5ab6edb26
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:50:58 2022 +0530

f1 added

commit f32b9469d954ae7e4b2082b9f14435494697f390
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:17:14 2022 +0530

m2 added

commit 09ac428c3a0b18850aec1c2f611898d0e42ab706
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:16:24 2022 +0530

m1 added

```

Fig. Shows Complete history on the master branch of feature latest commit  
(Merge branch “feature”)

We can't go back to previous commit

**D:\one drive data\Desktop\Branches>git reset --hard HEAD~3**

The screenshot shows the VS Code interface with the Terminal tab selected. The terminal window displays the following text:

```
Merge branch 'feature'

commit 76e4b63e5cb83abb78c0f5e1861f2e1c11b4754a (feature)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:51:35 2022 +0530

    f2 added

commit fb7ff2718002060f6dc5d541f9c635b5ab6edbb26
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:50:58 2022 +0530

    f1 added

commit f32b9469d954ae7e4b2082b9f14435494697f390
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:17:14 2022 +0530

    m2 added

commit 09ac428c3a0b18850aec1c2f611898d0e42ab706
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:16:24 2022 +0530

    m1 added

D:\one drive data\Desktop\Branches>git reset --hard HEAD~3
fatal: ambiguous argument 'HEAD~3': unknown revision or path not in the working tree.
Use '--' to separate paths from revisions, like this:
'git <command> [<revision>...] -- [<file>...]'
```

Fig. Fatal Error

To resolve this error, we need to undo the latest commit only:

D:\one drive data\Desktop\Branches>git reset --hard HEAD~1

The screenshot shows the VS Code interface with the Terminal tab selected. The terminal window displays the following text:

```
D:\one drive data\Desktop\Branches>git reset --hard HEAD~1
HEAD is now at f32b946 m2 added

D:\one drive data\Desktop\Branches>git log
commit f32b9469d954ae7e4b2082b9f14435494697f390 (HEAD -> master)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:17:14 2022 +0530

    m2 added

commit 09ac428c3a0b18850aec1c2f611898d0e42ab706
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:16:24 2022 +0530

    m1 added
```

Fig. Error resolved

The screenshot shows the VS Code interface with the terminal tab selected. The terminal window displays the following command and its output:

```

Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date: Fri May 20 23:16:24 2022 +0530

m1 added

D:\one\drive\data\Desktop\Branches>git switched feature
git: 'switched' is not a git command. See 'git --help'.

D:\one\drive\data\Desktop\Branches>git switch feature
Switched to branch 'feature'

D:\one\drive\data\Desktop\Branches>git log
commit 76e4b63e5cb83abb78c0f5e1861f2e1c11b4754a (HEAD -> feature)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date: Fri May 20 23:51:35 2022 +0530

f2 added

commit fb7ff2718002060f6dc5d541f9c635b5ab6edb26
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date: Fri May 20 23:50:58 2022 +0530

f1 added

```

Fig. feature switched commit history is same in current branch

Let's Switch to master and create m3.txt file

The screenshot shows the VS Code interface with the terminal tab selected. The terminal window displays the following command and its output:

```

D:\one\drive\data\Desktop\Branches>git switch master
Switched to branch 'master'

D:\one\drive\data\Desktop\Branches>git add .

D:\one\drive\data\Desktop\Branches>git commit -m "m3 added"
[master 07e9d09] m3 added
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 master/m3.txt

D:\one\drive\data\Desktop\Branches>]

```

Fig. Successful created m3.txt file

The screenshot shows the VS Code interface with the terminal tab selected. The terminal window displays the following command and its output:

```

D:\one\drive\data\Desktop\Branches>git merge feature
Merge made by the 'ort' strategy.
 feature/f1.txt | 0
 feature/f2.txt | 0
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 feature/f1.txt
 create mode 100644 feature/f2.txt

D:\one\drive\data\Desktop\Branches>]

```

Fig. Successful merge feature branch into master

The screenshot shows the VS Code interface with the 'TERMINAL' tab selected. The terminal window displays the command 'git log' and its output, which shows the full commit history of the 'master' branch. The commits are as follows:

- commit 7c39bb6e630d4f972c76ed54b8e64d47311e9eef (HEAD -> master)  
Merge: 07e9d09 76e4b63  
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>  
Date: Sat May 21 01:09:28 2022 +0530
- Merge branch 'feature'
- commit 07e9d09bc9a7726060ec9d2cc0685058c45d78ce  
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>  
Date: Sat May 21 01:07:36 2022 +0530
- m3 added
- commit 76e4b63e5cb83abb78c0f5e1c11b4754a (feature)  
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>  
Date: Fri May 20 23:51:35 2022 +0530
- f2 added
- commit fb7ff2718002060f6dc541f9c635b5ab6edb26  
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>  
Date: Fri May 20 23:50:58 2022 +0530
- f1 added
- commit f32b9469d954ae7e4b2082b9f14435494697f390  
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>  
Date: Fri May 20 23:17:14 2022 +0530
- m2 added

The left sidebar shows the 'EXPLORER', 'OPEN EDITORS', 'OUTLINE', and 'TIMELINE' sections. The 'OPEN EDITORS' section shows three files: m1.txt, m2.txt, and m3.txt, with m3.txt currently selected.

Fig. full commit history shows in master branch

## Let's do again –squash

The screenshot shows the VS Code interface with the 'TERMINAL' tab selected. The terminal window displays the command 'git reset --hard HEAD~1' and its output, which shows the switch to the previous commit. The commit history now starts from the previous commit:

- D:\one drive data\Desktop\Branches>git reset --hard HEAD~1  
HEAD is now at 07e9d09 m3 added
- D:\one drive data\Desktop\Branches>git log  
commit 07e9d09bc9a7726060ec9d2cc0685058c45d78ce (HEAD -> master)  
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>  
Date: Sat May 21 01:07:36 2022 +0530
- m3 added
- commit f32b9469d954ae7e4b2082b9f14435494697f390  
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>  
Date: Fri May 20 23:17:14 2022 +0530
- m2 added
- commit 09ac428c3a0b18850aec1c2f611898d0e42ab706  
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>  
Date: Fri May 20 23:16:24 2022 +0530
- m1 added

The left sidebar shows the 'EXPLORER', 'OPEN EDITORS', 'OUTLINE', and 'TIMELINE' sections. The 'OPEN EDITORS' section shows three files: m1.txt, m2.txt, and m3.txt, with m3.txt currently selected.

Fig. switch to previous commit

The screenshot shows the VS Code interface with the terminal tab selected. The terminal window displays the following command and its output:

```
D:\one\drive\data\Desktop\Branches>git merge --squash feature
Automatic merge went well; stopped before committing as requested
Squash commit -- not updating HEAD

D:\one\drive\data\Desktop\Branches>git log
commit 07e9d09bc9a7726060ec9d2cc0685058c45d78ce (HEAD -> master)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Sat May 21 01:07:36 2022 +0530

    m3 added

commit f32b9469d954ae7e4b2082b9f14435494697f390
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:17:14 2022 +0530

    m2 added

commit 09ac428c3a0b18850aec1c2f611898d0e42ab706
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:16:24 2022 +0530

    m1 added

D:\one\drive\data\Desktop\Branches>
```

Fig. squash is used to squash the previous commits into one

The screenshot shows the VS Code interface with the terminal tab selected. The terminal window displays the following command and its output:

```
D:\one\drive\data\Desktop\Branches>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:  feature/f1.txt
    new file:  feature/f2.txt

D:\one\drive\data\Desktop\Branches>git commit -m "Master and Feature Merged"
[master 6b3a969] Master and Feature Merged
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 feature/f1.txt
 create mode 100644 feature/f2.txt

D:\one\drive\data\Desktop\Branches>git status
On branch master
nothing to commit, working tree clean

D:\one\drive\data\Desktop\Branches>
```

Fig. Committed Master and Feature Merge

```

D:\one drive data\Desktop\Branches>git log
commit 6b3a9693656f0dc1dcc1294e2e1b0f188494259d (HEAD -> master)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Sat May 21 01:19:56 2022 +0530

    Master and Feature Merged

commit 07e9d09bc9a7726060ec9d2cc0685058c45d78ce
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Sat May 21 01:07:36 2022 +0530

    m3 added

commit f32b9469d954ae7e4b2082b9f14435494697f390
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:17:14 2022 +0530

    m2 added

commit 09ac428c3a0b18850aec1c2f611898d0e42ab706
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:16:24 2022 +0530

    m1 added

D:\one drive data\Desktop\Branches>

```

Fig. Success Merged in Commit History

## Rebasing Theory:

Rebasing is a process to reapply commits on top of another base trip.

It is used to apply a sequence of commits from distinct branches into a final commit.

It is an alternative of git merge command

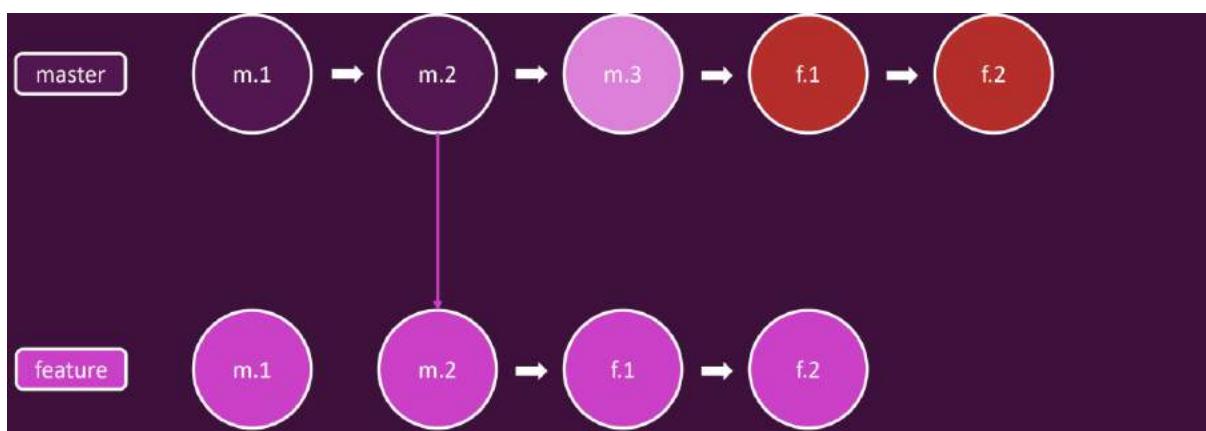


Fig. Shows Master & Feature of rebase

## What Happens? When we use rebase??

From a content perspective, rebasing is changing the base of your branch from one commit to another making it appear as if you'd created your branch from a different commit.

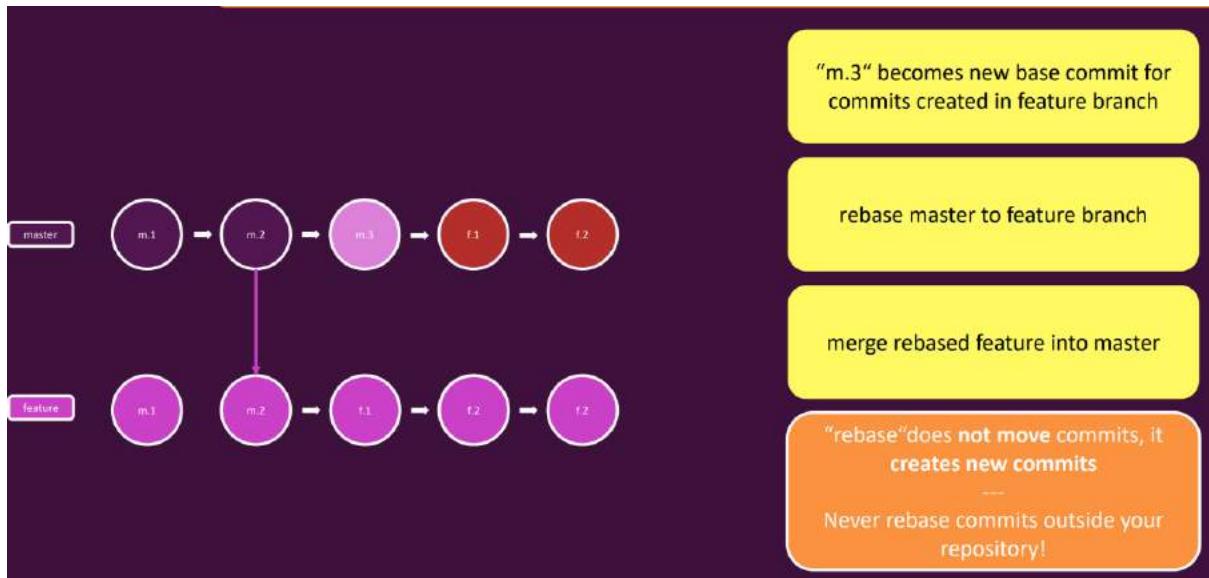


Fig. Shows using of rebase

## Let's start rebasing

```
D:\one drive data\Desktop\Branches>git log
commit 797897e5dd3039dca14fba1ec0166799e23a9275 (HEAD -> master)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Sat May 21 17:55:04 2022 +0530

    m3 added

commit 4fb0560cc444aeb50c835a2b2cf7eb955bc32651
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Sat May 21 17:42:55 2022 +0530

    m2 added

commit e334ac365705a0c4b4cce713e2307df8438eabe8
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Sat May 21 17:41:45 2022 +0530

    m1 added
```

Fig. Exactly same structure as shows rebase of master as above referenced

```

D:\One Drive Data\Desktop\Branches>git log
commit 782e699a12faed17535a192e72b8fe1bd595de (HEAD -> feature)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Sat May 21 17:46:30 2022 +0530

    f2 added

commit 36fc7615d0b1740b6eb2ccfa3c3f428ae92eeb8
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Sat May 21 17:44:13 2022 +0530

    f1 added

commit 4fb0560cc444aeb50c835a2b2cf7eb955bc32651
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Sat May 21 17:42:55 2022 +0530

    m2 added

commit e334ac365705a0c4b4cce713e2307df8438eabe8
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Sat May 21 17:41:45 2022 +0530

    m1 added

```

Fig. Exactly same structure as shows rebases of feature as above referenced

## Applying rebase in feature branch

```

D:\One Drive Data\Desktop\Branches>git rebase master
Successfully rebased and updated refs/heads/feature.

D:\One Drive Data\Desktop\Branches>git log
commit 5e29bfff5c3ffe06d05db7f7f1565e296b415380 (HEAD -> feature)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Sat May 21 17:46:30 2022 +0530

    f2 added

commit 8a9a06cb85bd9339a4dd335d227324887c35fbc3
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Sat May 21 17:44:13 2022 +0530

    f1 added

commit 797897e5dd3039dc14fba1ec0166799e23a9275 (master)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Sat May 21 17:55:04 2022 +0530

    m3 added

commit 4fb0560cc444aeb50c835a2b2cf7eb955bc32651
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Sat May 21 17:42:55 2022 +0530

    m2 added

commit e334ac365705a0c4b4cce713e2307df8438eabe8
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Sat May 21 17:41:45 2022 +0530

```

Fig. Successful applied rebase in feature branch to merge master commits.

**Remember Note: after applying rebase commit history ID changed**

The screenshot shows a terminal window with the following output:

```
D:\one\drive\data\Desktop\Branches>git merge feature
Updating 797897e..5e29bff
Fast-forward
 feature/f1.txt | 0
 feature/f2.txt | 0
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 feature/f1.txt
 create mode 100644 feature/f2.txt
D:\one\drive\data\Desktop\Branches>
```

Fig. fast-forward merging

## When to apply rebase?

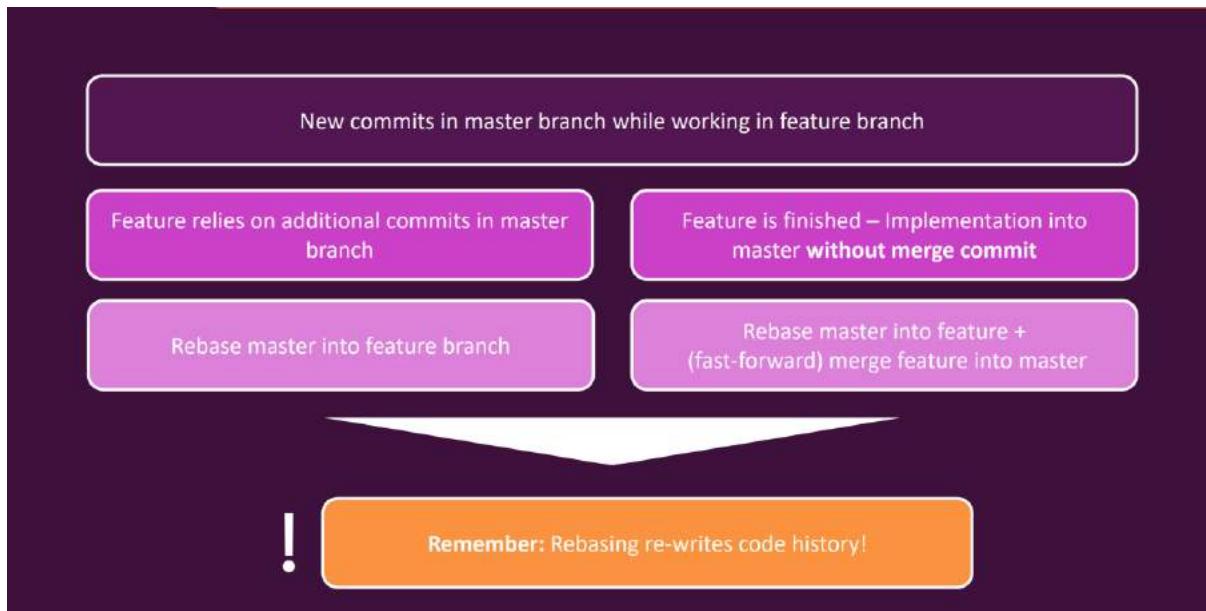


Fig. Workflow

**Clever Note: Never use rebase when project is dependent to others branches.**

## Handling Merge Conflicts:

How to fix conflicts during the merge?

Scenario: If two people in two different branches work on same file in the end

The screenshot shows a terminal window with the following content:

```
feature > f1.txt
1 This is the change made in master branch ! Creating first feature

D:\one drive data\Desktop\Branches>git branch
  feature
* master

D:\one drive data\Desktop\Branches>git add .

D:\one drive data\Desktop\Branches>git commit -m "Change feature1 in master"
[master 274c14c] Change feature1 in master
  1 file changed, 1 insertion(+)

D:\one drive data\Desktop\Branches>
```

Fig. Shows Working feature in Master Branch

The screenshot shows a terminal window with the following content:

```
feature > f1.txt
1 This is the change made in feature branch ! Creating first feature

D:\one drive data\Desktop\Branches>git switch feature
Switched to branch 'feature'

D:\one drive data\Desktop\Branches>git add .

D:\one drive data\Desktop\Branches>git commit -m " Change features in feature branch"
On branch feature
nothing to commit, working tree clean

D:\one drive data\Desktop\Branches>
```

Fig. Shows Working feature in Feature Branch

The screenshot shows the VS Code interface with the following details:

- EXPLORER**: Shows files m1.txt, m2.txt, m3.txt in the master branch, and f1.txt in the feature branch.
- OPEN EDITORS**: Displays the content of f1.txt.
- BRANCHES**: Shows the current branch is feature, with f1.txt selected.
- EDITOR**: Shows a merge conflict in f1.txt. The conflict is between HEAD (Current Change) and the feature branch. The content is:
 

```

feature > f1.txt
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
1 <<<<< HEAD (Current Change)
2 This is the change made in master branch ! Creating first feature
3 =====
4 This is the change made in feature branch ! Creating first feature
5 >>>>> feature (Incoming Change)
6
      
```
- TERMINAL**: Shows the command output:
 

```

D:\one drive data\Desktop\Branches>git branch
  feature
* master

D:\one drive data\Desktop\Branches>git merge feature
Auto-merging feature/f1.txt
CONFLICT (content): Merge conflict in feature/f1.txt
Automatic merge failed; fix conflicts and then commit the result.

D:\one drive data\Desktop\Branches>
      
```

Fig. Shows Conflicts Created

## How to handle these conflicts?

The screenshot shows the VS Code interface with the following details:

- EXPLORER**: Shows files m1.txt, m2.txt, m3.txt in the master branch, and f1.txt in the feature branch.
- OPEN EDITORS**: Displays the content of f1.txt.
- BRANCHES**: Shows the current branch is master, with f1.txt selected.
- EDITOR**: Shows the content of f1.txt with the same conflict as before.
- TERMINAL**: Shows the command output:
 

```

D:\one drive data\Desktop\Branches>git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
    both modified:  feature/f1.txt

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

D:\one drive data\Desktop\Branches>
      
```

Fig. git status shows how to handle conflicts

The screenshot shows a VS Code interface with the following details:

- EXPLORER**: Shows files m1.txt, m2.txt, m3.txt, f1.txt (highlighted in red), f2.txt, and a folder named feature.
- OPEN EDITORS**: Displays the content of f1.txt, which contains two conflicting changes:

```
feature > f1.txt
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
1 <<<<< HEAD (Current Change)
2 This is the change made in master branch ! Creating first feature
3 =====
4 This is the change made in feature branch ! Creating first feature
5 >>>>> feature (Incoming Change)
```
- BRANCHES**: Shows branches feature (selected) and master.
- PROBLEMS**, **OUTPUT**, **DEBUG CONSOLE**, **TERMINAL**: The terminal shows the following output:

```
D:\one\drive\data\Desktop\Branches>git merge feature
Auto-merging feature/f1.txt
CONFLICT (content): Merge conflict in feature/f1.txt
Automatic merge failed; fix conflicts and then commit the result.

D:\one\drive\data\Desktop\Branches>git log --merge
commit 409837b77d3b174f5bd10a5d6672f26d4e6b21fd (feature)
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date:   Sat May 21 18:46:10 2022 +0530

    Changes features in feature branch

commit 274c14c6c285597fdb16499fd942e0376b1e343f (HEAD -> master)
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date:   Sat May 21 18:35:29 2022 +0530

    Change feature1 in master
```
- TIMELINE**: Shows the history of changes.

Fig. `git log --merges` show how to handle conflicts

The screenshot shows the VS Code interface with the following details:

- EXPLORER**: Shows files m1.txt, m2.txt, m3.txt, f1.txt (marked as modified), and f2.txt.
- OPEN EDITORS**: Shows f1.txt (marked as modified).
- BRANCHES**: Shows branches feature (selected) and master.
- OUTLINE**: Shows no symbols found in document 'f1.txt'.
- EDITOR**: Displays the content of f1.txt with a merge conflict:

```
feature > f1.txt
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
1 <<<<< HEAD (Current Change)
2 This is the change made in master branch ! Creating first feature
3 =====
4 This is the change made in feature branch ! Creating first feature
5 >>>>> feature (Incoming Change)
```
- TERMINAL**: Shows the command output of `git diff` and the file content.

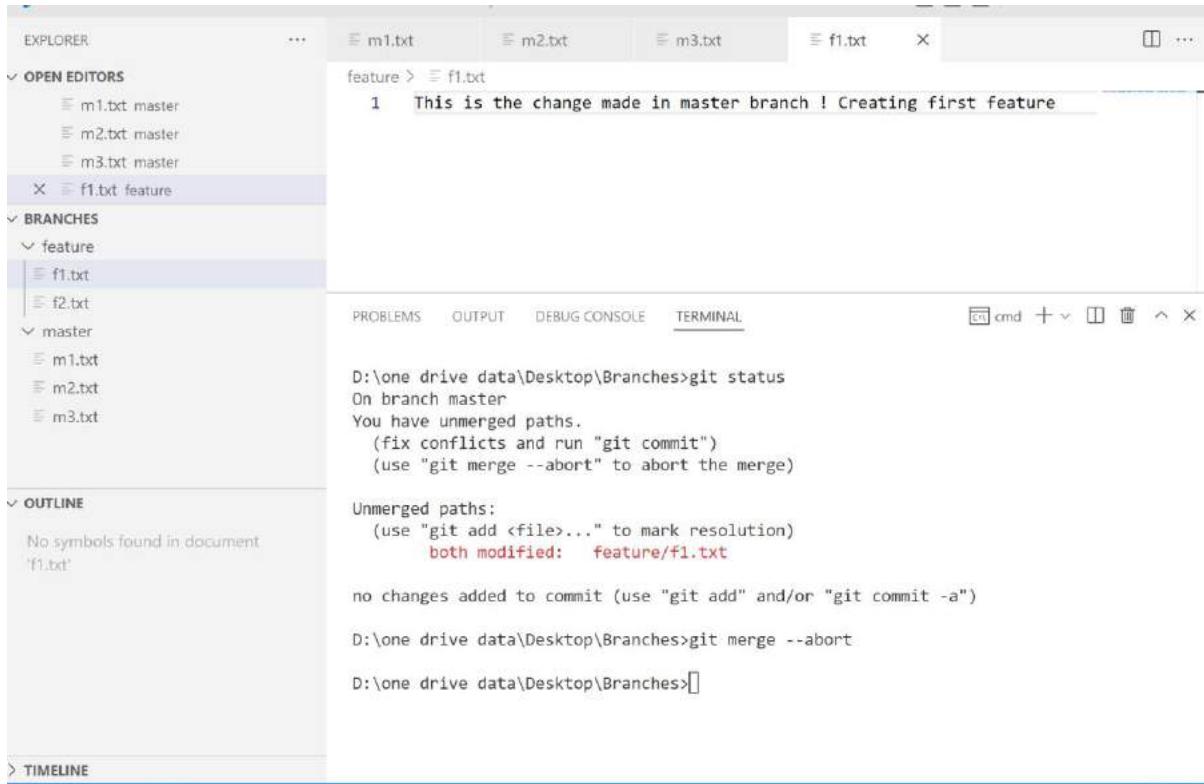
Detailed Terminal Output:

```
D:\one\drive\data\Desktop\Branches>git diff
diff --cc feature/f1.txt
index b4b8add,82fbfd67..0000000
--- a/feature/f1.txt
+++ b/feature/f1.txt
@@@ -1,1 -1,1 +1,5 @@@
- This is the change made in master branch ! Creating first feature
- This is the change made in feature branch ! Creating first feature
+<<<<< HEAD
++This is the change made in master branch ! Creating first feature
+=====
++This is the change made in feature branch ! Creating first feature
++>>>>> feature
```

D:\one\drive\data\Desktop\Branches>

**Fig. git diff** show how to handle conflicts

- Use git-reset or git merge --abort to cancel a merge that had conflicts



The screenshot shows a code editor interface with the following details:

- EXPLORER:** Shows files m1.txt, m2.txt, m3.txt under the master branch, and f1.txt, f2.txt under the feature branch.
- OPEN EDITORS:** Displays three tabs: m1.txt, m2.txt, and m3.txt (disabled), and one active tab f1.txt with the content: "1 This is the change made in master branch ! Creating first feature".
- BRANCHES:** Shows the feature branch containing f1.txt and f2.txt, and the master branch containing m1.txt, m2.txt, and m3.txt.
- PROBLEMS:** Shows a warning about unmerged paths in f1.txt.
- OUTPUT:** Shows the command D:\one drive data\Desktop\Branches>git status followed by output indicating unmerged paths in f1.txt.
- DEBUG CONSOLE:** Shows the command D:\one drive data\Desktop\Branches>git merge --abort.
- TERMINAL:** Shows the command D:\one drive data\Desktop\Branches>.
- COMMANDS:** Includes icons for cmd, terminal, file operations, and others.

Fig. git merge --abort to cancel a merge that had conflicts

- Accepting Current Change in VS code and resolving the issue:

The screenshot shows the VS Code interface with the following details:

- EXPLORER:** Shows files m1.txt, m2.txt, m3.txt in the master branch, and f1.txt in the feature branch.
- OPEN EDITORS:** 1 UNSAVED (f1.txt)
- BRANCHES:** feature (f1.txt, f2.txt), master (m1.txt, m2.txt, m3.txt).
- PROBLEMS:** A red error icon is shown above the terminal.
- TERMINAL:**

```
D:\One Drive Data\Desktop\Branches>git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
    both modified:  feature/f1.txt

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

D:\One Drive Data\Desktop\Branches>git add .

D:\One Drive Data\Desktop\Branches>git commit -m "merged feature and master in f1 file"
[master 1755d3f] merged feature and master in f1 file

D:\One Drive Data\Desktop\Branches>
```

Fig. Merged current changes in master branch

The screenshot shows the VS Code interface with the following details:

- EXPLORER:** Shows files m1.txt, m2.txt, m3.txt in the master branch, and f1.txt in the feature branch.
- OPEN EDITORS:** 1 UNSAVED (f1.txt)
- BRANCHES:** feature (f1.txt, f2.txt), master (m1.txt, m2.txt, m3.txt).
- TERMINAL:**

```
D:\One Drive Data\Desktop\Branches>git log
commit 1755d3f086d5384f3d4a3a1309a7ae4d498946cc (HEAD -> master)
Merge: 274c14c 409837b
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date: Sat May 21 19:05:13 2022 +0530

merged feature and master in f1 file

commit 409837b77d3b174f5bd10a5d6672f26d4e6b21fd (feature)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date: Sat May 21 18:46:10 2022 +0530

Changes features in feature branch

commit 274c14c6c285597fdb16499fd942e0376b1e343f
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date: Sat May 21 18:35:29 2022 +0530

Change feature1 in master

commit 5e29bfff5c3ffe06d05db7f7f1565e296b415380
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date: Sat May 21 17:46:30 2022 +0530

f2 added

commit 8a9a06cb85bd9339a4dd335d227324887c35fb3
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date: Sat May 21 17:44:13 2022 +0530

f1 added
```

Fig.History committed in master branch

## Understanding “git cherry-pick”

The command `git cherry-pick` is typically used to introduce particular commits from one branch within a repository onto a different branch. A common use is to forward- or back-port commits from a maintenance branch to a development branch.

## Merge vs Rebase vs Cherry Pick



Fig. Quick recap

**Scenario: A person is working on two different branches and typo mistake in code then we have added specific commit to branch HEAD.**

A screenshot of a terminal window within a code editor interface (VS Code). The terminal shows the following command sequence:

```
D:\one drive data\Desktop\Branches>git branch
  feature
* master

D:\one drive data\Desktop\Branches>git commit -m "Working on m1"
[master bc75b55] Working on m1
  1 file changed, 1 insertion(+)

D:\one drive data\Desktop\Branches>
```

The terminal output indicates that a commit was made to the `master` branch with the message "Working on m1". The commit hash is `bc75b55`. The commit message contains a typo ("Important" instead of "Important").

Fig. Working m1 file of master and typo mistake in Important

The screenshot shows the VS Code interface with the following details:

- EXPLORER** sidebar: Shows branches `feature`, `master`, and an `OUTLINE` section indicating no symbols found in `f4.txt`.
- OPEN EDITORS**: 1 UNSAVED tab bar: `m1.txt`, `f3.txt`, `f4.txt` (active), `m2.txt`, `m3.txt`.
- TERMINAL** tab bar: `cmd`, `+ new`, `copy`, `ctrl`, `ctrl +`, `ctrl -`, `ctrl +`, `ctrl -`, `ctrl +`, `ctrl -`.
- Terminal Output:**

```
D:\one drive data\Desktop\Branches>git switch feature
Switched to branch 'feature'

D:\one drive data\Desktop\Branches>git add .
[feature 26513b2] f3 added
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 feature/f3.txt

D:\one drive data\Desktop\Branches>git commit -m "f4 added"
[feature 214c6f7] f4 added
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 feature/f4.txt

D:\one drive data\Desktop\Branches>
```

Fig. Building Features `f3.txt` and `f4 .txt` in feature Branch

Now Suddenly, we realised their typo in code of `m1` file we need to fix it

The screenshot shows the VS Code interface with the following details:

- EXPLORER** sidebar: Shows branches `feature`, `master`, and an `OUTLINE` section.
- OPEN EDITORS**: 1 UNSAVED tab bar: `m1.txt` (active), `f3.txt`, `f4.txt`, `m2.txt`, `m3.txt`.
- TERMINAL** tab bar: `cmd`, `+ new`, `copy`, `ctrl`, `ctrl +`, `ctrl -`, `ctrl +`, `ctrl -`.
- Terminal Output:**

```
master > m1.txt
1 Some Important thing in master

D:\one drive data\Desktop\Branches>git add .

D:\one drive data\Desktop\Branches>git commit -m "typo in m1 file added"
[feature b7f6257] typo in m1 file added
 1 file changed, 1 insertion(+), 1 deletion(-)

D:\one drive data\Desktop\Branches>
```

Fig. Shows Fix typo in `m1` master file

The screenshot shows the VS Code interface with the following details:

- EXPLORER** view: Shows branches and files. The 'feature' branch contains files f1.txt, f2.txt, f3.txt, and f4.txt. The 'master' branch contains files m1.txt, m2.txt, and m3.txt.
- TERMINAL** view: Displays the commit history for the 'feature' branch. It includes commits for typo fixes (f1.txt, f2.txt, f3.txt) and a merge commit that adds f4.txt to the master branch.
- OUTPUT** view: Shows the output of the command 'git log'.

```

D:\one\drive\data\Desktop\Branches>git log
commit b7f6257adf5941db359f0d5dd99890533a1c40d0 (HEAD -> feature)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Sat May 21 19:47:01 2022 +0530

    typo in m1 file added

commit 47b817c854c8395deb775e5befa828c7aa6577da
Merge: 214c6f7 bc75b55
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Sat May 21 19:46:02 2022 +0530

    Merge branch 'master' into feature

commit 214c6f77ad09e731c5afff61e70334c45a22c9ef
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Sat May 21 19:36:48 2022 +0530

    f4 added

commit 26513b263f20c9c804b1d4a3b90480fe159f1ae0
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Sat May 21 19:35:59 2022 +0530

    f3 added

commit bc75b559f861ce06572a990834d36d2f7836a255 (master)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Sat May 21 19:32:52 2022 +0530
  
```

> TIMELINE Working on m1

Fig. shows commit history of feature branch

**Problem:** Now, if we merge feature branch into master then all commits will be going to add. But I want merge the specific typo commit of feature branch into master branch.

To resolve the problem introduced “git cherry-pick”

The screenshot shows the VS Code interface with the following details:

- EXPLORER** view: Shows branches and files. The 'feature' branch contains files f1.txt, f2.txt, f3.txt, and f4.txt. The 'master' branch contains files m1.txt, m2.txt, and m3.txt.
- TERMINAL** view: Shows the command 'git cherry-pick' being run to add the specific commit from the feature branch into the master branch.

```

D:\one\drive\data\Desktop\Branches>git switch master
Switched to branch 'master'

D:\one\drive\data\Desktop\Branches>git cherry-pick b7f6257adf5941db359f0d5dd99890533a1c40d0
[master 9620022]  typo in m1 file added
Date: Sat May 21 19:47:01 2022 +0530
1 file changed, 1 insertion(+), 1 deletion(-)

D:\one\drive\data\Desktop\Branches>
  
```

Fig. specific commits has been added to master branch

The screenshot shows a code editor interface with a terminal tab active. The terminal window displays a git log output. The log shows several commits:

- commit 9620022a114860b8d4abaa047d3ea7b7c92b537e (HEAD -> master)**  
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>  
Date: Sat May 21 19:47:01 2022 +0530  
typo in m1 file added
- commit bc75b559f861ce06572a990834d36d2f7836a255**  
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>  
Date: Sat May 21 19:32:52 2022 +0530  
Working on m1
- commit 1755d3f086d5384f3d4a3a1309a7ae4d498946cc**  
Merge: 274c14c 409837b  
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>  
Date: Sat May 21 19:05:13 2022 +0530  
merged feature and master in f1 file
- commit 409837b77d3b174f5bd10a5d6672f26d4e6b21fd**  
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>  
Date: Sat May 21 18:46:10 2022 +0530  
Changes features in feature branch
- commit 274c14c6c285597fdb16499fd942e0376b1e343f**  
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>  
Date: Sat May 21 18:35:29 2022 +0530

Fig. git log shows that commit is successful.

### Master branch:

```
D:\one drive data\Desktop\Branches>git log
commit 9620022a114860b8d4abaa047d3ea7b7c92b537e (HEAD -> master)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date: Sat May 21 19:47:01 2022 +0530
```

typo in m1 file added

### Feature branch:

```
commit b7f6257adf5941db359f0d5dd99890533a1c40d0 (HEAD -> feature)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date: Sat May 21 19:47:01 2022 +0530
```

typo in m1 file added

Clever Note: copies commit is with new ID because of “git cherry-pick”

## Working with “git tag”:

- Tags are ref's that point to specific points in Git history
- Git supports two types of tags: lightweight and annotated.
- Light weighted: it's just a pointer to a specific commit.
- Annotated: stored as full objects in the Git database

## Lightweight tags:

The screenshot shows a terminal window within a code editor interface. The terminal output is as follows:

```
D:\one\drive\data\Desktop\Branches>git tag
D:\one\drive\data\Desktop\Branches>git tag 2.2 b7f6257adf5941db359f0d5dd99890533a1c40d0
D:\one\drive\data\Desktop\Branches>git show 2.2
commit b7f6257adf5941db359f0d5dd99890533a1c40d0 (HEAD -> feature, tag: 2.2)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Sat May 21 19:47:01 2022 +0530

    typo in m1 file added

diff --git a/master/m1.txt b/master/m1.txt
index ad9c9e6..86150c3 100644
--- a/master/m1.txt
+++ b/master/m1.txt
@@ -1 +1 @@
-Some Important thing in master
\ No newline at end of file
+Some Important thing in master
\ No newline at end of file

D:\one\drive\data\Desktop\Branches>
```

Fig. Lightweight tags Example

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
cmd + ×

D:\one drive data\Desktop\Branches>git checkout 2.2
Note: switching to '2.2'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

git switch -c <new-branch-name>

Or undo this operation with:

git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at b7f6257  typo in m1 file added

D:\one drive data\Desktop\Branches>
```

Fig. Shows we can checkout with tags in detached head (lightweight tag)

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

cmd + ·

OPEN EDIT... 1 UNSAVED

m1.txt master  
  f3.txt feature  
  f4.txt feature  
m2.txt master  
m3.txt master  
● f1.txt feature

BRANCHES

feature  
  f1.txt

D:\one drive data\Desktop\Branches>git tag -d 2.2  
Deleted tag '2.2' (was b7f6257)

D:\one drive data\Desktop\Branches>git tag

D:\one drive data\Desktop\Branches>

Fig. Successful tag deleted

## Annotated tags:

stored as full objects in the Git database

The screenshot shows a terminal window with the following output:

```
D:\one\drive\data\Desktop\Branches>git tag -a 2.9 bc75b559f861ce06572a990834d36d2f7836a255 -m "Previous tags"
D:\one\drive\data\Desktop\Branches>git tag
2.9
D:\one\drive\data\Desktop\Branches>git show 2.9
tag 2.9
Tagger: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Sat May 21 20:33:37 2022 +0530

Previous tags

commit bc75b559f861ce06572a990834d36d2f7836a255 (tag: 2.9)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Sat May 21 19:32:52 2022 +0530

Working on m1

diff --git a/master/m1.txt b/master/m1.txt
index e69de29..ad9c9e6 100644
--- a/master/m1.txt
+++ b/master/m1.txt
@@ -0,0 +1 @@
+Some Important thing in master
\ No newline at end of file

D:\one\drive\data\Desktop\Branches>[]
```

Fig. Annotated tags

## Wrap Up Git Diving Deeper:

git stash	Temporary storage for unstaged and uncommitted changes
git reflog	A log of all project changes made including deleted commits
git merge	Combining commits from different branches by creating a new merge commit (recursive) or by moving the HEAD (fast-forward)
git rebase	Change the base (i.e. the parent commit) of commits in another branch
git cherry-pick	Copy commit including the changes made only in this commit as HEAD to other branch

Fig. Git Diving Deeper commands.

**From local to remote understanding:**



Fig. Git knowledge with GitHub in cloud

**Module Overview:**

- **What is GitHub? How Git and GitHub are connected?**
- **Remote Branches, Remote Tracking Branches & Local Tracking Branches**
- **Understanding Upstream and Git clone**

## What is GitHub?



Fig. shows about Git and GitHub.

## How Git and GitHub are connected?

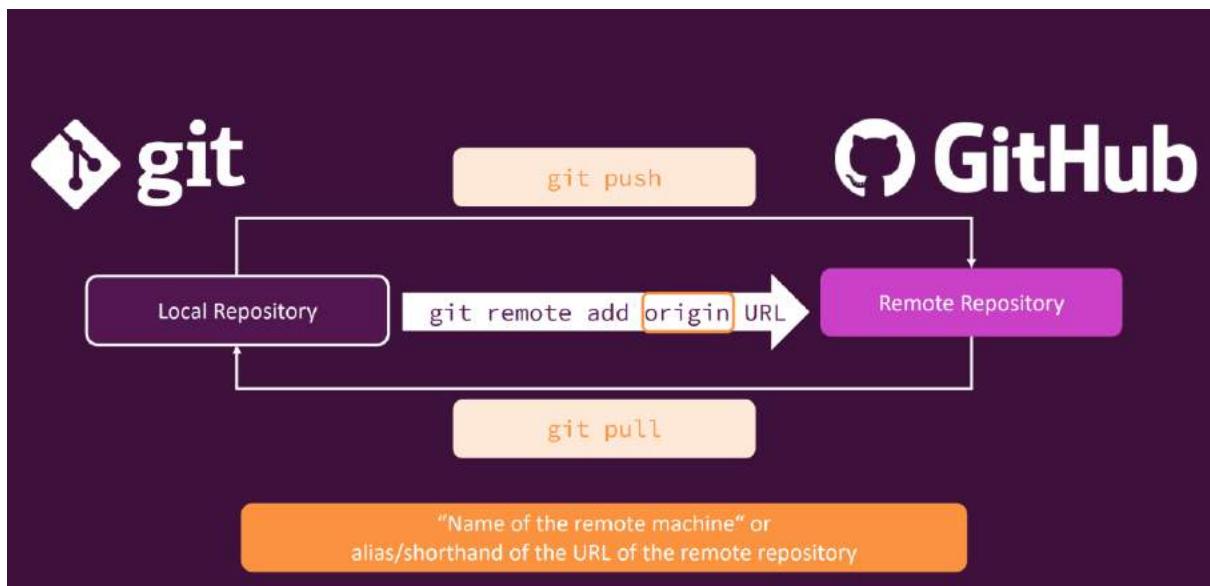


Fig. shows connecting git and GitHub - local to remote repository

## Creating a GitHub account and introducing GitHub

1. Open <https://github.com> in a web browser, and then select **Sign up**.

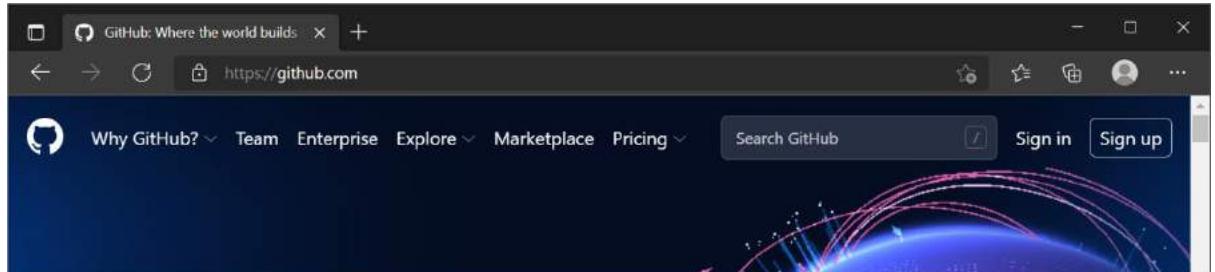


Fig. official site

2. Enter your email address.



Fig. Add your email address

3. **Create a password** for your new GitHub account, and **Enter a username**, too. Next, choose whether you want to receive updates and announcements via email, and then select **Continue**.

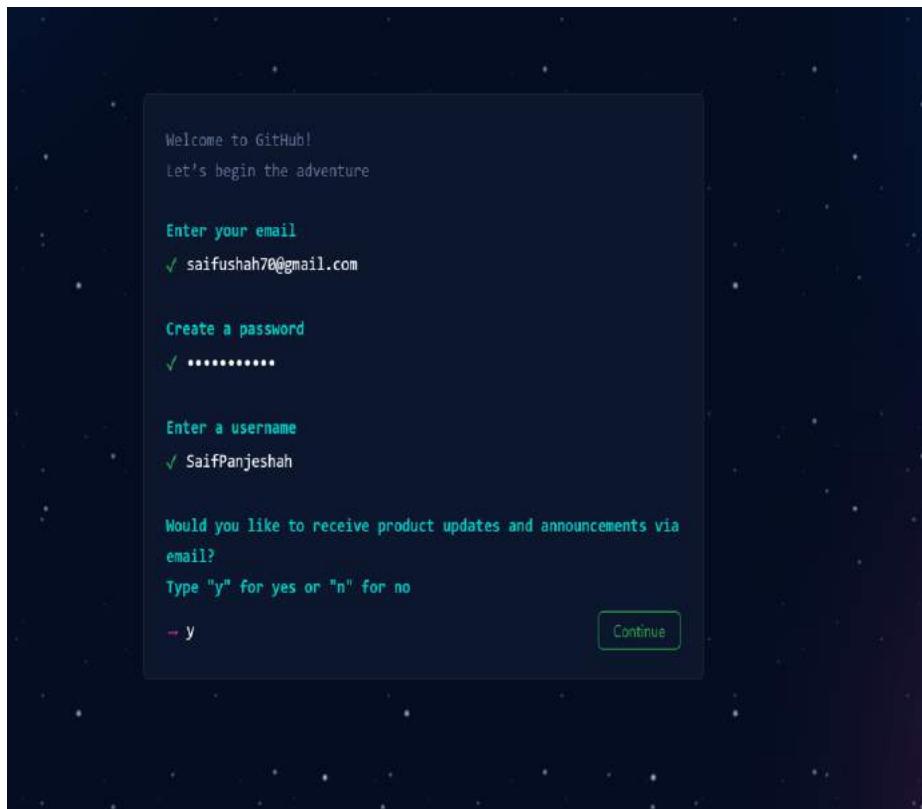


Fig. Creation of Username & Password

4. **Verify your account** by solving a puzzle. Select the **Start Puzzle** button to do so, and then follow the prompts.

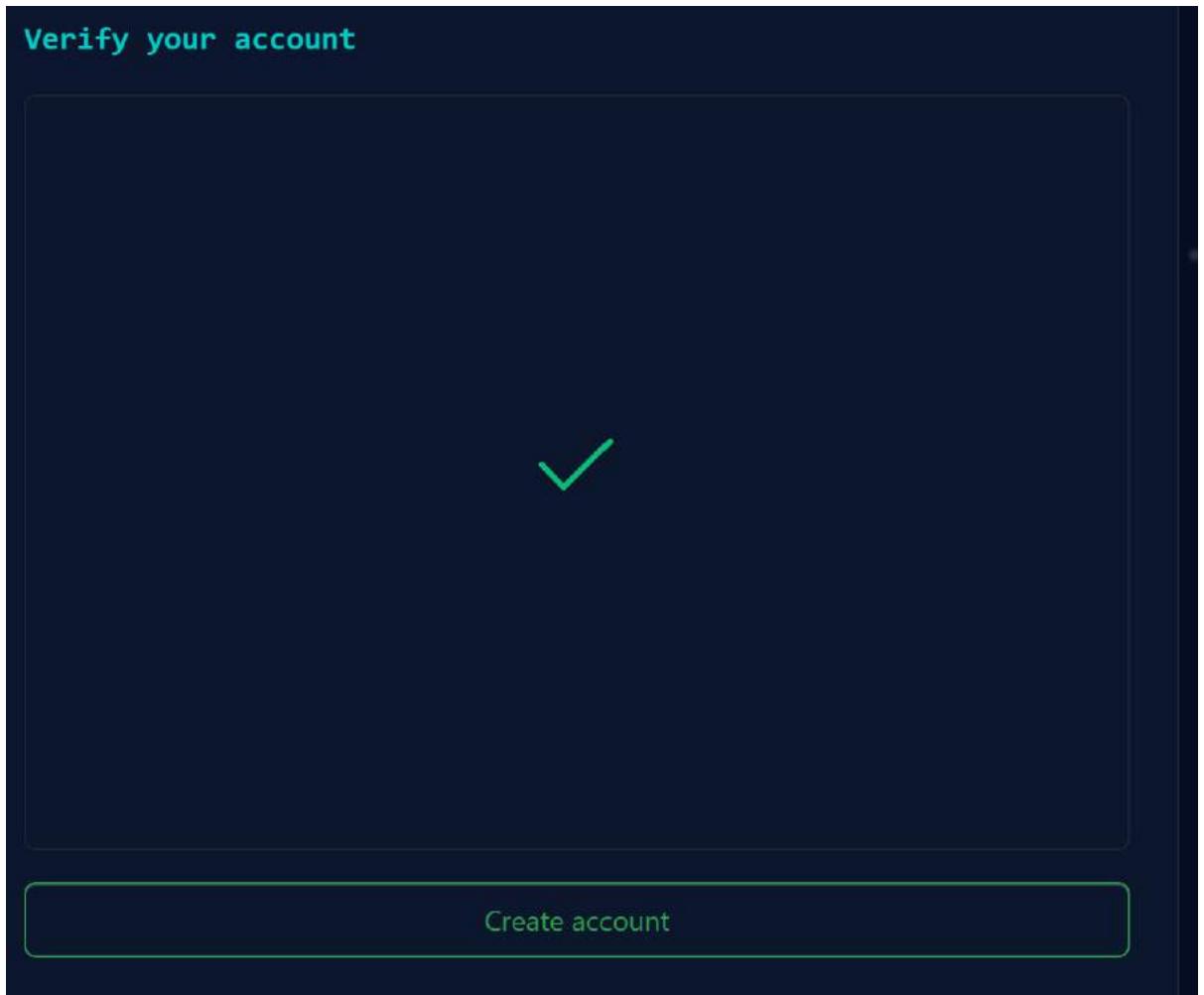


Fig. Account Verified

5. After you verify your account, select the **Create account** button.
6. Next, GitHub sends a launch code to your email address. Type that launch code in the **Enter code** dialog, and then press **Enter**.

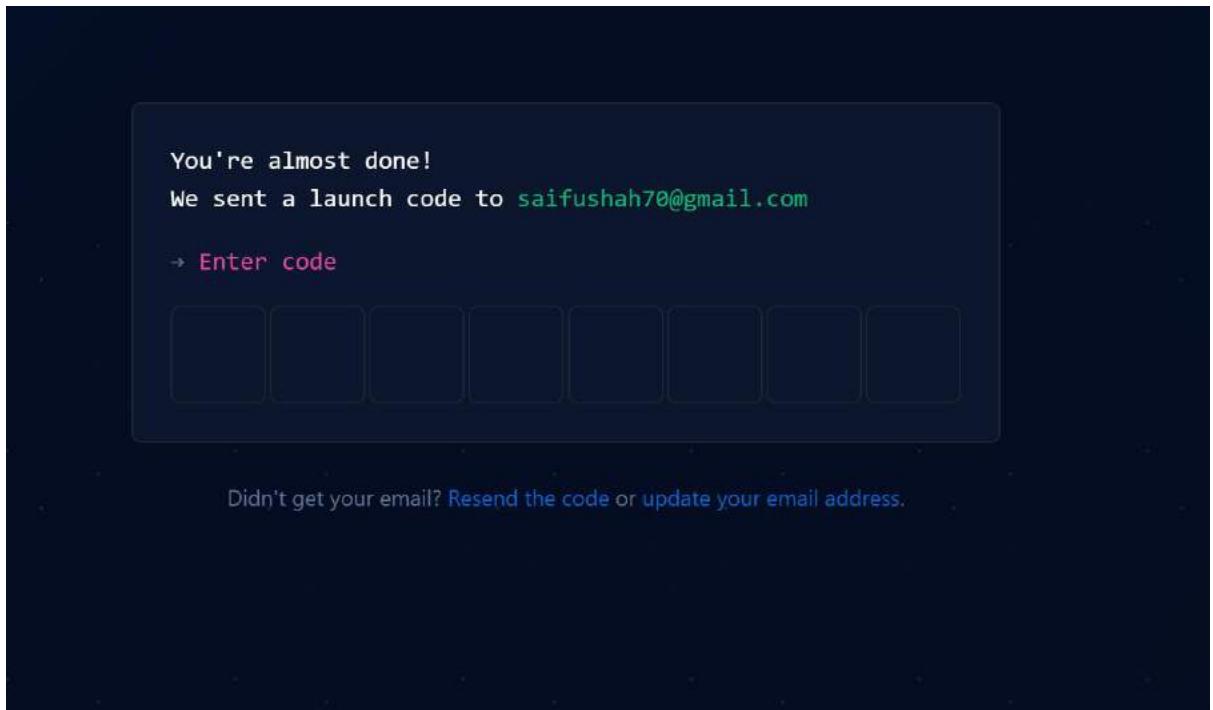


Fig. Enter the Code

7. GitHub asks you some questions to help tailor your experience. Choose the answers that apply to you in the following dialogs:
  - **How many team members will be working with you?**
  - **What specific features are you interested in using?**
8. On the **Where teams collaborate and ship** screen, you can choose whether you want to use the Free account or the Team account. To choose the **Free** account, select the **Skip personalization** button.

**Tip**

You can always upgrade your account later. See the [Types of GitHub accounts](#) page to learn more.

GitHub opens a personalized page in your browser.

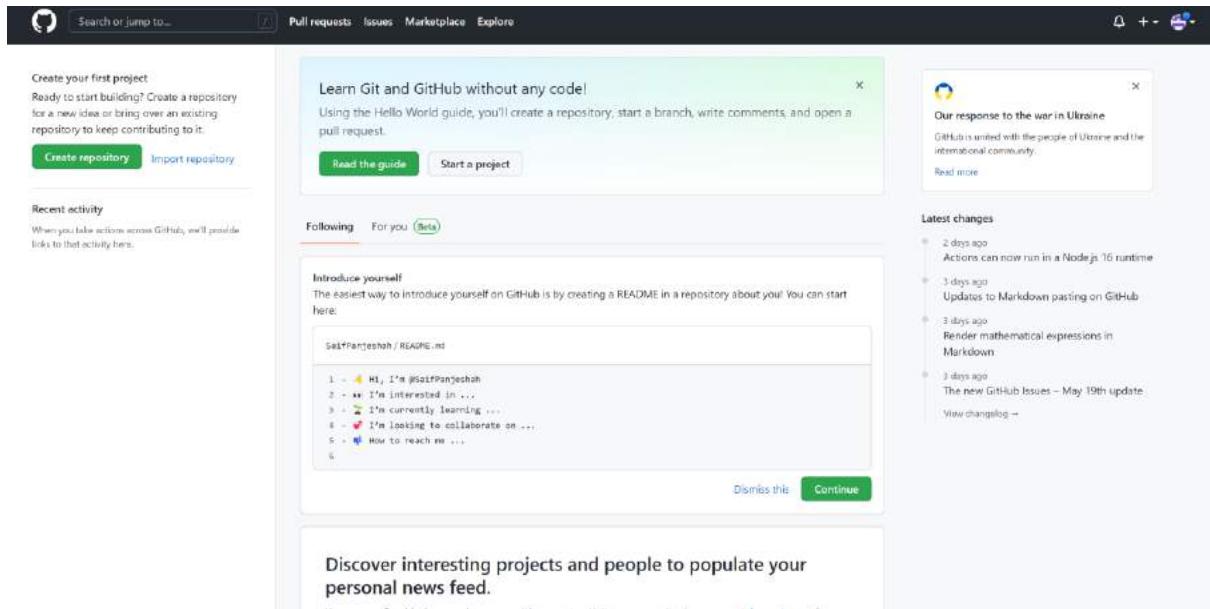


Fig. Account Creation Successful

## Creating a repository:

1. Click on create repository on top left corner in account or go to your profile and create repositories in account

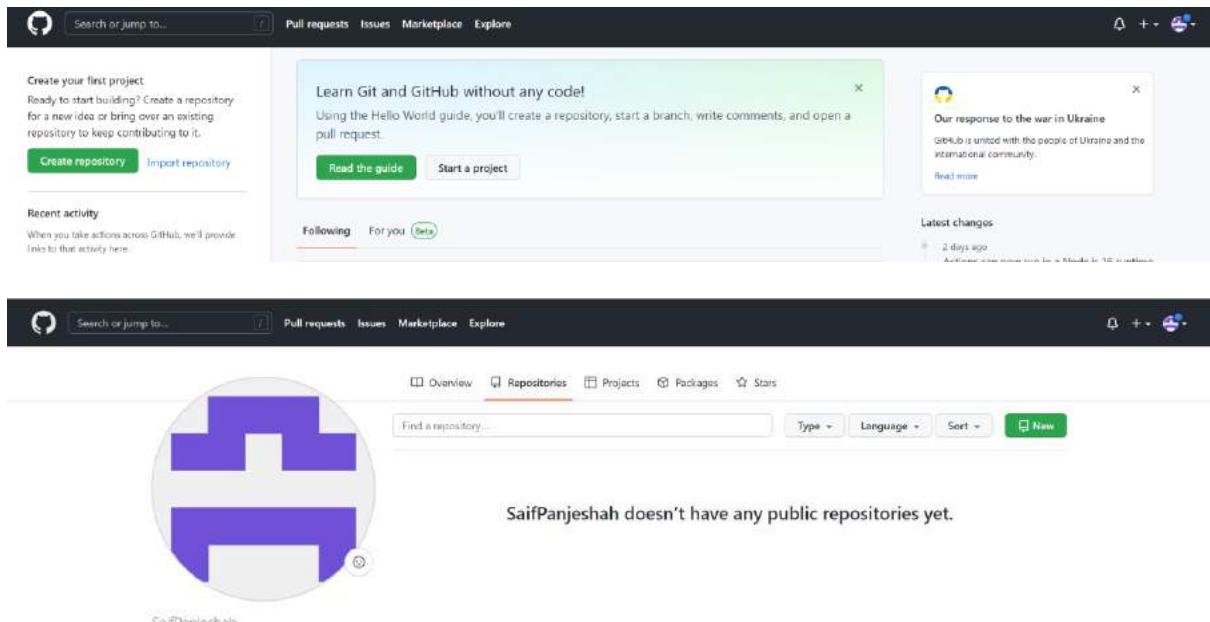


Fig. Creating repository

- Add your repository name and choose your access public and private Initialize your repository with README file , .gitignore not to choose track from list and choose license others Do's and not Do's .

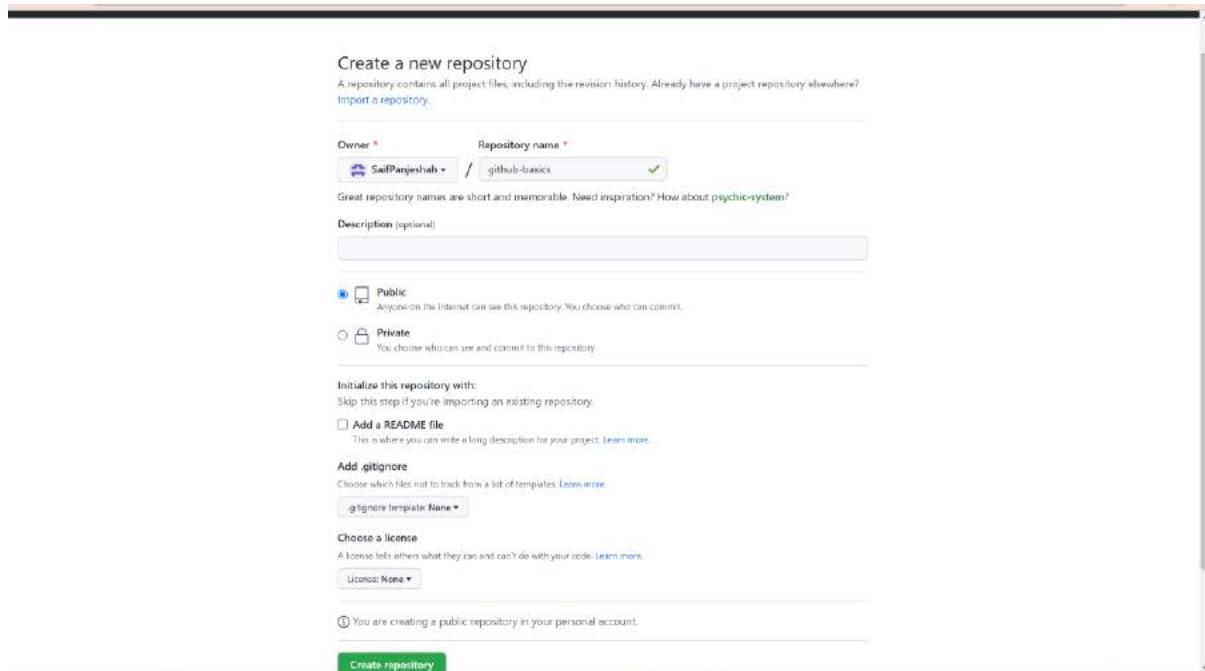


Fig. Created a new repository

- Quick Step-up to connect with your git account.

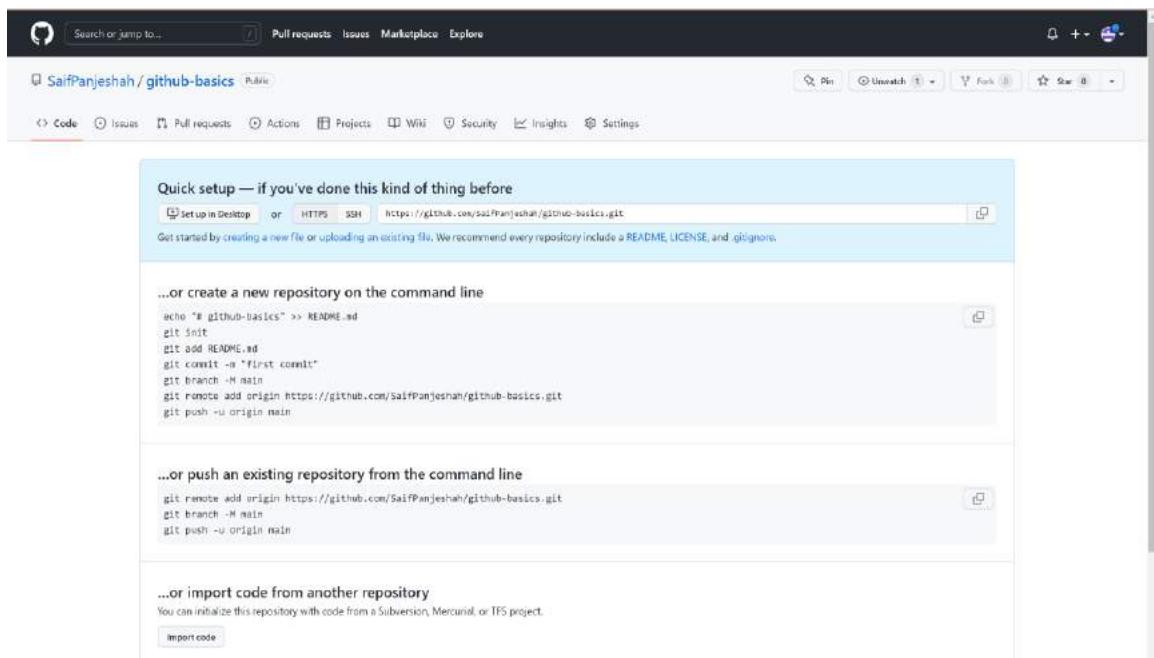


Fig. Setup git on GitHub

#### 4. Click on GitHub Home tab your repository has successful created



Fig. Repository Created

#### 5. Go to profile and check successful created as popular repository and contribution in the last year

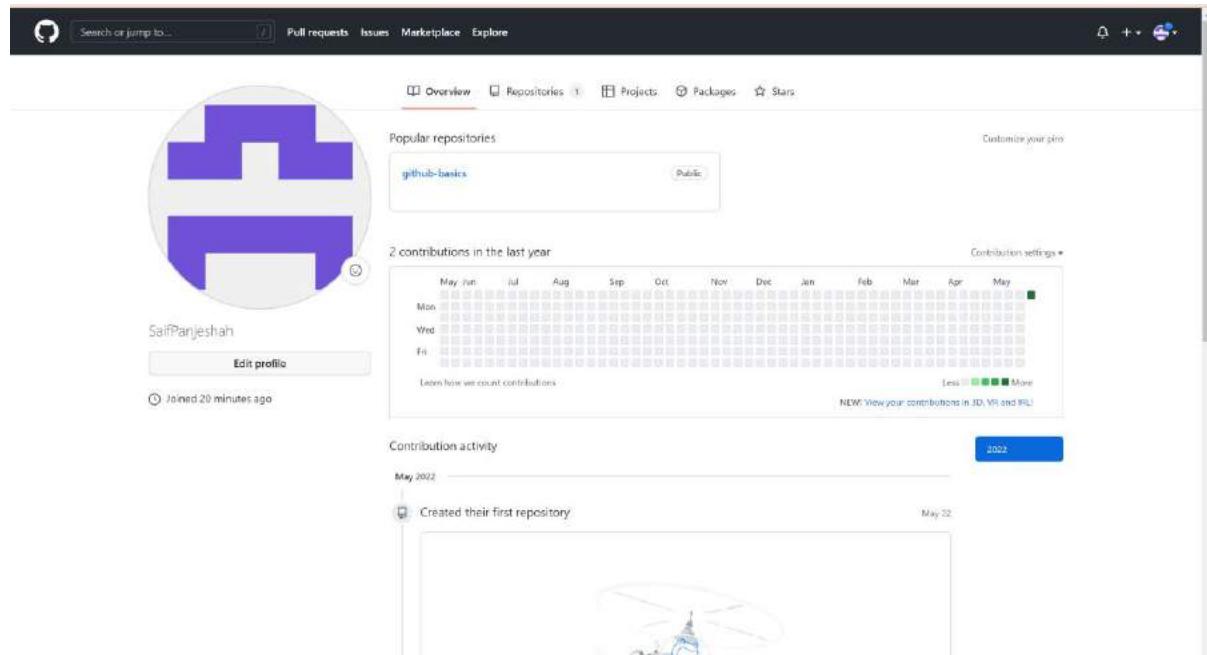


Fig. Successful remote repository added

## Connecting Local and remote repository:

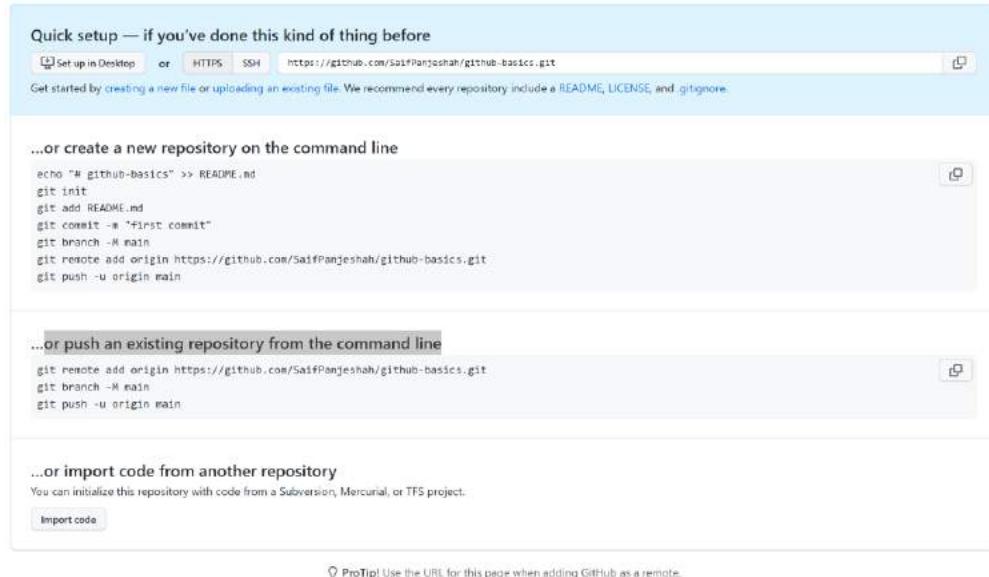


Fig. pushing repository from git

## Git local repository

The screenshot shows the VS Code interface with the 'GITHUB...' extension active. The Explorer sidebar shows a local repository named 'master' containing a file 'm1.txt'. The terminal at the bottom shows the command-line steps to initialize and commit the file.

```
D:\one\drive\data\Desktop\github-basics>git init
Initialized empty Git repository in D:/one\drive\data\Desktop\github-basics/.git/
D:\one\drive\data\Desktop\github-basics>git add .
D:\one\drive\data\Desktop\github-basics>git commit -m "m1 added"
[master (root-commit) b9fe76d] m1 added
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 master/m1.txt
D:\one\drive\data\Desktop\github-basics>
```

Fig. Creation of git repository

```
D:\one\drive\data\Desktop\github-basics>git branch
* master

D:\one\drive\data\Desktop\github-basics>git log
commit b9fe76d083da58314ad83e2fa13138d7dabf0294 (HEAD -> master)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Sun May 22 17:50:15 2022 +0530

    m1 added

D:\one\drive\data\Desktop\github-basics>
```

Fig. branches and commit history.

**Now, how we can move this local repository to Cloud?**

**Using Command:**

```
git remote add origin https://github.com/SaifPanjeshah/gitHub1-basics.git
//add the remote connection from local repository
```

```
git push origin master //Bring our local changes, our local information on
remote repository
```

A screenshot of the Visual Studio Code interface. The Explorer sidebar shows a repository named 'GITHUB1-BASICS' with a single branch 'master'. A file named 'm1.txt' is open in the editor, containing the number '1'. Below the editor are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', and 'TERMINAL'. The 'TERMINAL' tab is active, displaying a command-line session:

```
D:\one\drive\data\Desktop\GitHub1-basics>git remote add origin https://github.com/SaifPanjeshah/gitHub1-basics.git
D:\one\drive\data\Desktop\GitHub1-basics>git push origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 228 bytes | 228.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/SaifPanjeshah/gitHub1-basics.git
 * [new branch]      master -> master
D:\one\drive\data\Desktop\GitHub1-basics>
```

Fig. Successful push our local repository on GitHub

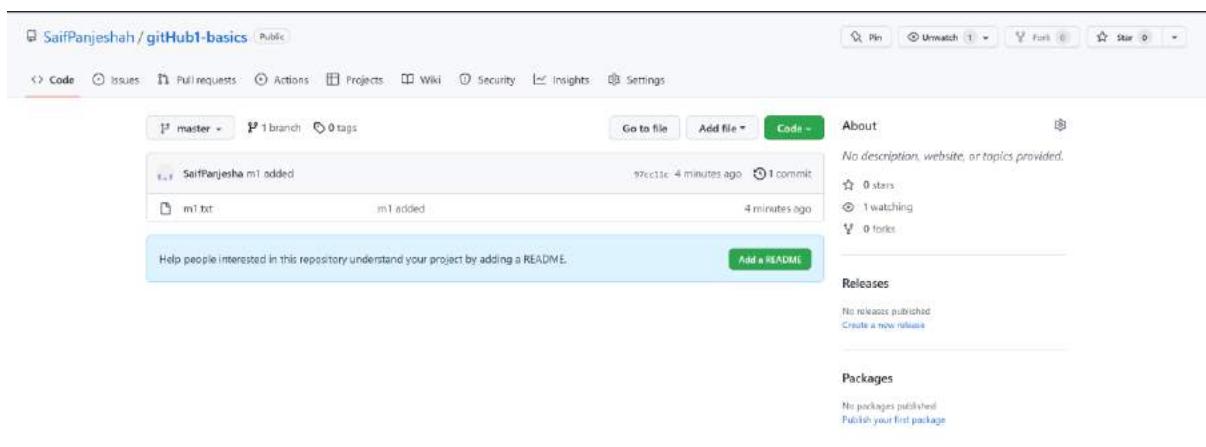


Fig. Successful push our local repository on GitHub via accessing through Internet browser.

**Now, how can we push our code with latest approach? Personal Access Token?**

## Understanding the Personal Access Token:

### Creating a token:

- a. [Verify your email address](#), if it hasn't been verified yet.
- b. In the upper-right corner of any page, click your profile photo, then click **Settings**.

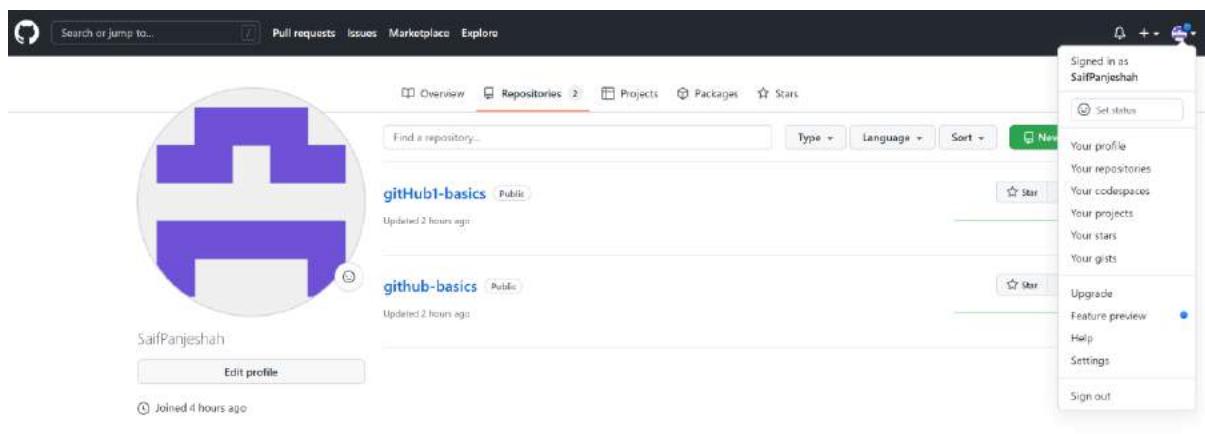


Fig. After verifying email address GitHub profile

- c. In the left sidebar, click **Developer settings**

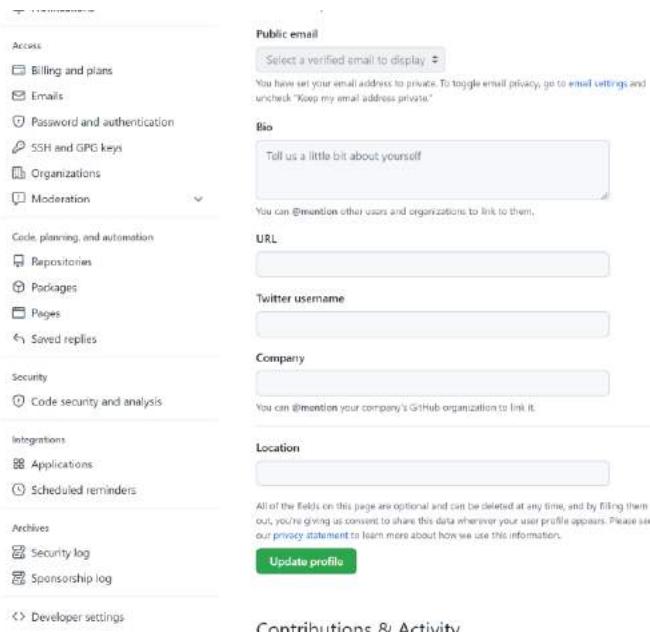
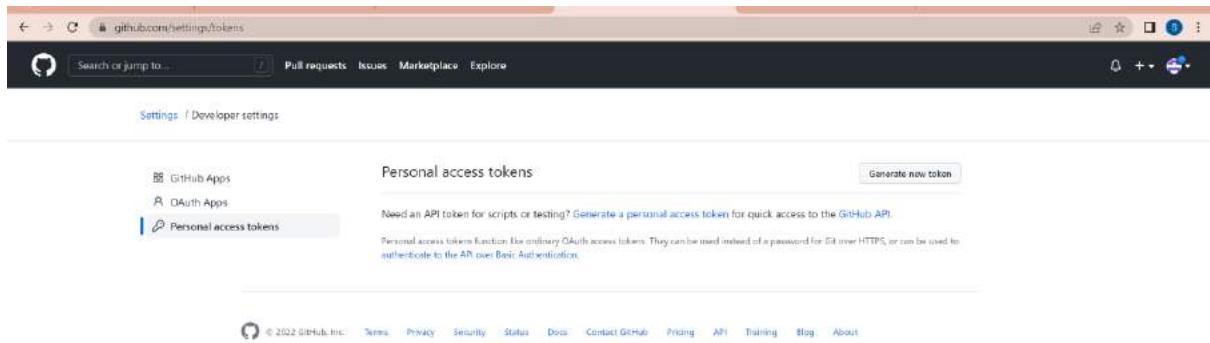


Fig. Setting to open developer setting

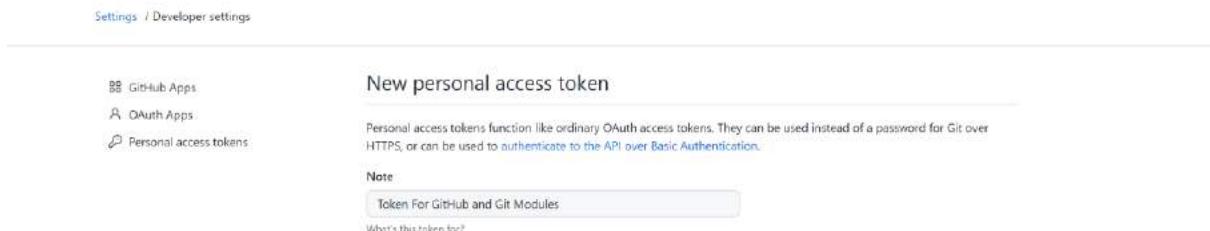
d. In the left sidebar, click **Personal access tokens**.



The screenshot shows the GitHub settings interface. The URL in the address bar is `github.com/settings/tokens`. The left sidebar has links for "GitHub Apps", "OAuth Apps", and "Personal access tokens", with "Personal access tokens" being the active tab. The main content area is titled "Personal access tokens" and contains a sub-instruction: "Need an API token for scripts or testing? Generate a personal access token for quick access to the GitHub API." Below this, there is a note: "Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication." At the top right of this section is a button labeled "Generate new token". The footer of the page includes links for Terms, Privacy, Security, Status, Docs, Contact GitHub, Pricing, API, Training, Blog, and About.

Fig. Personal access tokens

e. Give your token a descriptive name.



The screenshot shows the "New personal access token" creation page. The URL in the address bar is `github.com/settings/tokens/new`. The left sidebar shows "GitHub Apps", "OAuth Apps", and "Personal access tokens". The main content area is titled "New personal access token" and contains a note: "Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication." Below this is a "Note" section with a text input field containing "Token For GitHub and Git Modules". A question "What's this token for?" is visible below the note section.

Fig. Descriptive Name of Token

f. To give your token an expiration, select the **Expiration** drop-down menu, then click a default or use the calendar picker.



The screenshot shows a dropdown menu for "Expiration" with the value "90 days" selected. To the right of the dropdown, a message states: "The token will expire on Sat, Aug 20 2022".

Fig. Expiration days

- g. Select the scopes, or permissions, you'd like to grant this token. To use your token to access repositories from the command line, select **repo**.

#### Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)

<input checked="" type="checkbox"/> <b>repo</b>	Full control of private repositories
<input type="checkbox"/> repo:status	Access commit status
<input type="checkbox"/> repo_deployment	Access deployment status
<input type="checkbox"/> public_repo	Access public repositories
<input type="checkbox"/> repo:invite	Access repository invitations
<input type="checkbox"/> security_events	Read and write security events
<input checked="" type="checkbox"/> <b>workflow</b>	Update GitHub Action workflows
<input checked="" type="checkbox"/> <b>write:packages</b>	Upload packages to GitHub Package Registry
<input type="checkbox"/> read:packages	Download packages from GitHub Package Registry
<input checked="" type="checkbox"/> <b>delete:packages</b>	Delete packages from GitHub Package Registry
<input type="checkbox"/> <b>admin:org</b>	Full control of orgs and teams, read and write org projects
<input type="checkbox"/> write:org	Read and write org and team membership, read and write org projects
<input type="checkbox"/> read:org	Read org and team membership, read org projects
<input type="checkbox"/> <b>admin:public_key</b>	Full control of user public keys
<input type="checkbox"/> write:public_key	Write user public keys
<input type="checkbox"/> read:public_key	Read user public keys
<input type="checkbox"/> <b>admin:repo_hook</b>	Full control of repository hooks
<input type="checkbox"/> write:repo_hook	Write repository hooks
<input type="checkbox"/> read:repo_hook	Read repository hooks
<input type="checkbox"/> <b>admin:org_hook</b>	Full control of organization hooks
<input checked="" type="checkbox"/> <b>gist</b>	Create gists
<input checked="" type="checkbox"/> <b>notifications</b>	Access notifications
<input checked="" type="checkbox"/> <b>user</b>	Update ALL user data
<input type="checkbox"/> read:user	Read ALL user profile data

Fig. Permission for tokens excepts admin

#### h. Click **Generate token**.

The screenshot shows a list of GitHub API permissions with checkboxes. Most checkboxes are checked, except for 'admin:enterprise' and 'admin:gpg\_key'. At the bottom are two buttons: 'Generate token' (green) and 'Cancel'.

<input checked="" type="checkbox"/> notifications	Access notifications
<input checked="" type="checkbox"/> user	Update ALL user data
<input type="checkbox"/> read:user	Read ALL user profile data
<input type="checkbox"/> user:email	Access user email addresses (read-only)
<input type="checkbox"/> user:follow	Follow and unfollow users
<input checked="" type="checkbox"/> delete_repo	Delete repositories
<input checked="" type="checkbox"/> write:discussion	Read and write team discussions
<input type="checkbox"/> read:discussion	Read team discussions
<input type="checkbox"/> admin:enterprise	Full control of enterprises
<input type="checkbox"/> manage_runners:enterprise	Manage enterprise runners and runner-groups
<input type="checkbox"/> manage_billing:enterprise	Read and write enterprise billing data
<input type="checkbox"/> read:enterprise	Read enterprise profile data
<input type="checkbox"/> admin:gpg_key	Full control of public user GPG keys (Developer Preview)
<input type="checkbox"/> write:gpg_key	Write public user GPG keys
<input type="checkbox"/> read:gpg_key	Read public user GPG keys

**Generate token**   **Cancel**

**Fig. Generating Token**

The screenshot shows the 'Personal access tokens' section of the GitHub settings. It displays a single token with a copy icon and a 'Delete' button. A note at the bottom explains what personal access tokens are used for.

**Personal access tokens**

Tokens you have generated that can be used to access the [GitHub API](#).

Make sure to copy your personal access token now. You won't be able to see it again!

✓ ghp\_wd8vty043CDHAK71UIka0nDZLU80jB0T6k1N

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

[Generate new token](#) [Revoke all](#)

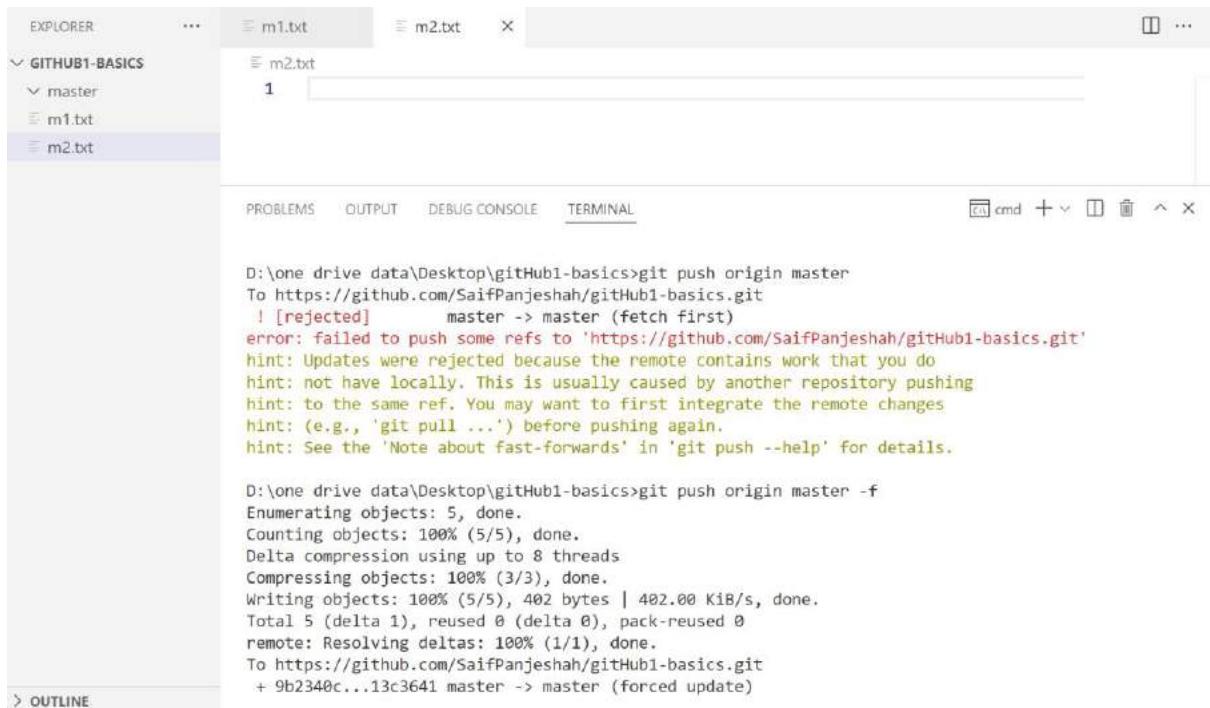
**Fig. Token Successful Generated**

## Pushing a Second Commit



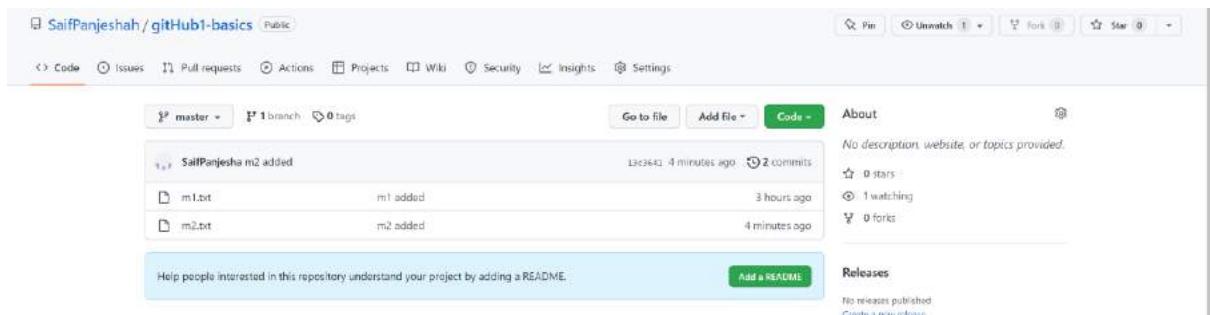
```
D:\one\drive\data\Desktop\gitHub1-basics>git add .  
D:\one\drive\data\Desktop\gitHub1-basics>git commit -m "m2 added"  
[master 13c3641] m2 added  
1 file changed, 0 insertions(+), 0 deletions(-)  
create mode 100644 m2.txt
```

Fig. Created M2 file



```
D:\one\drive\data\Desktop\gitHub1-basics>git push origin master  
To https://github.com/SaifPanjeshah/gitHub1-basics.git  
! [rejected]      master -> master (fetch first)  
error: failed to push some refs to 'https://github.com/SaifPanjeshah/gitHub1-basics.git'  
hint: Updates were rejected because the remote contains work that you do  
hint: not have locally. This is usually caused by another repository pushing  
hint: to the same ref. You may want to first integrate the remote changes  
hint: (e.g., 'git pull ...') before pushing again.  
hint: See the 'Note about fast-forwards' in 'git push --help' for details.  
  
D:\one\drive\data\Desktop\gitHub1-basics>git push origin master -f  
Enumerating objects: 5, done.  
Counting objects: 100% (5/5), done.  
Delta compression using up to 8 threads  
Compressing objects: 100% (3/3), done.  
Writing objects: 100% (5/5), 402 bytes | 402.00 KiB/s, done.  
Total 5 (delta 1), reused 0 (delta 0), pack-reused 0  
remote: Resolving deltas: 100% (1/1), done.  
To https://github.com/SaifPanjeshah/gitHub1-basics.git  
+ 9b2340c...13c3641 master -> master (forced update)
```

Fig. Push M2 file to remote repository



SaifPanjeshah / gitHub1-basics Public

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

master 1 branch 0 tags

Go to file Add file Code About

No description, website, or topics provided.

SaifPanjeshah m2 added 13c3641 4 minutes ago 2 commits

m1.txt m1 added 3 hours ago

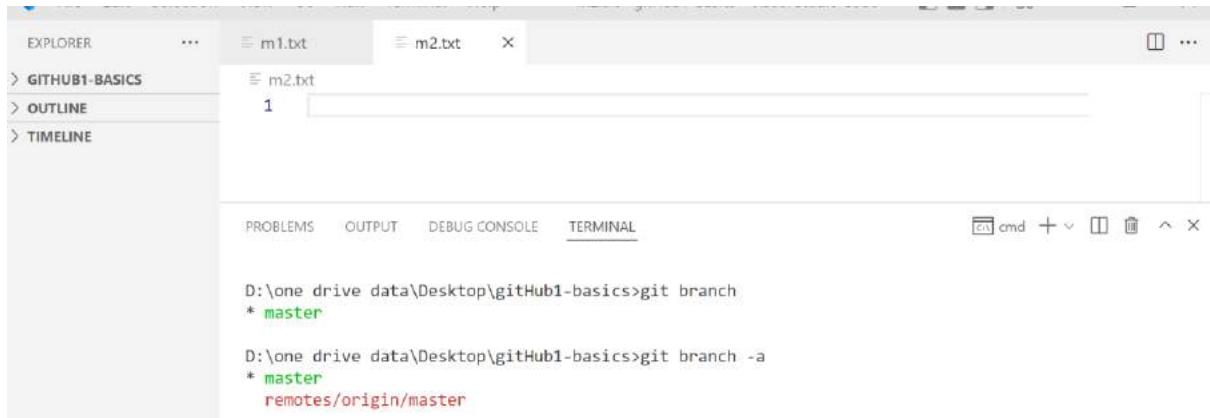
m2.txt m2 added 4 minutes ago

Help people interested in this repository understand your project by adding a README. Add a README

Releases No releases published Create a new release

Fig. Successful on GitHub

## From Local to remote Understanding the Workflow



A screenshot of a terminal window in a code editor. The terminal shows the command `git branch` being run twice. The first run shows a local `master` branch. The second run shows both the local `master` branch and its corresponding remote tracking branch `remotes/origin/master`. The terminal interface includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL, and a bottom bar with icons for cmd, +, and file operations.

```
D:\one\drive\data\Desktop\gitHub1-basics>git branch
* master

D:\one\drive\data\Desktop\gitHub1-basics>git branch -a
* master
  remotes/origin/master
```

Fig. added remote tracking branch

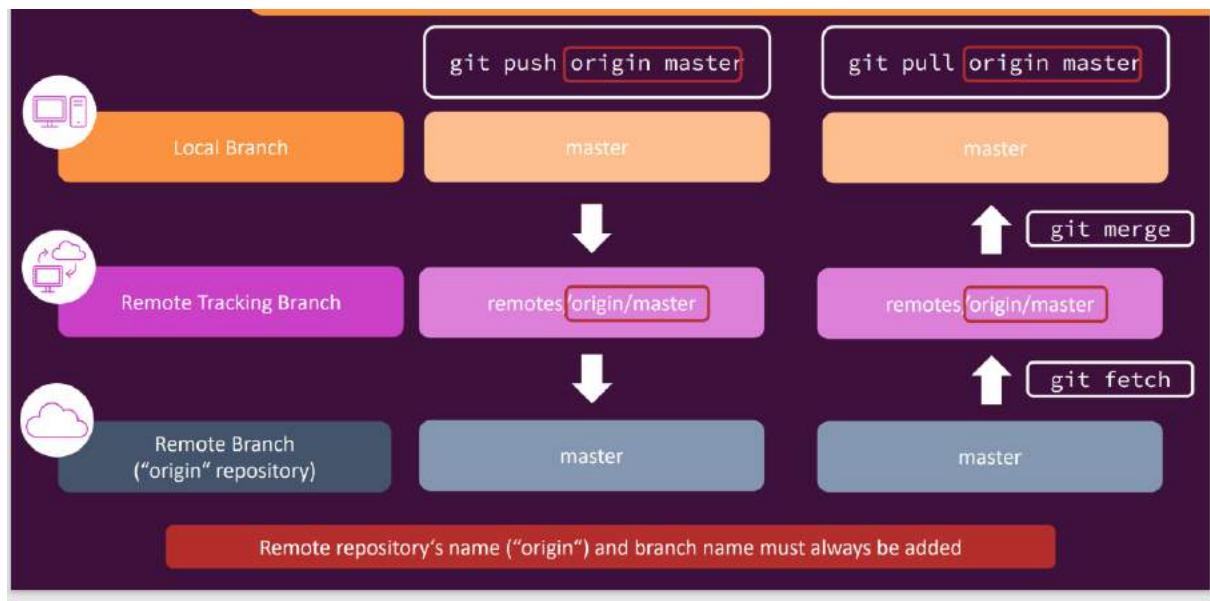
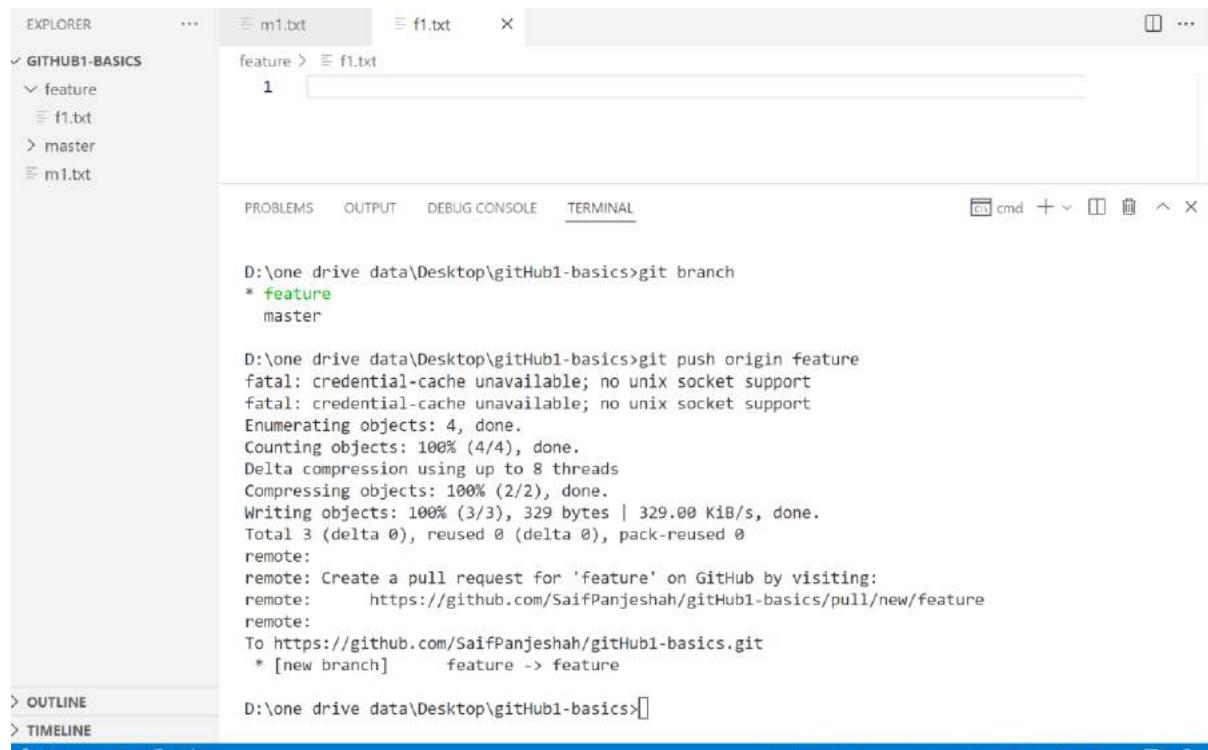


Fig. Local to remote Workflow

## Remote Tracking Branches in Practices:



The screenshot shows the VS Code interface. In the Explorer sidebar, there is a folder named 'GITHUB1-BASICS' containing 'feature', 'f1.txt', 'master', and 'm1.txt'. In the center workspace, there is a file named 'f1.txt' with the content '1'. Below the workspace are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL. The TERMINAL tab is active and displays the following command-line session:

```
D:\one drive data\Desktop\gitHub1-basics>git branch
* feature
  master

D:\one drive data\Desktop\gitHub1-basics>git push origin feature
fatal: credential-cache unavailable; no unix socket support
fatal: credential-cache unavailable; no unix socket support
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 329 bytes | 329.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'feature' on GitHub by visiting:
remote:   https://github.com/SaifPanjeshah/gitHub1-basics/pull/new/feature
remote:
To https://github.com/SaifPanjeshah/gitHub1-basics.git
 * [new branch]      feature -> feature

D:\one drive data\Desktop\gitHub1-basics>
```

Fig. Created New Feature Branch

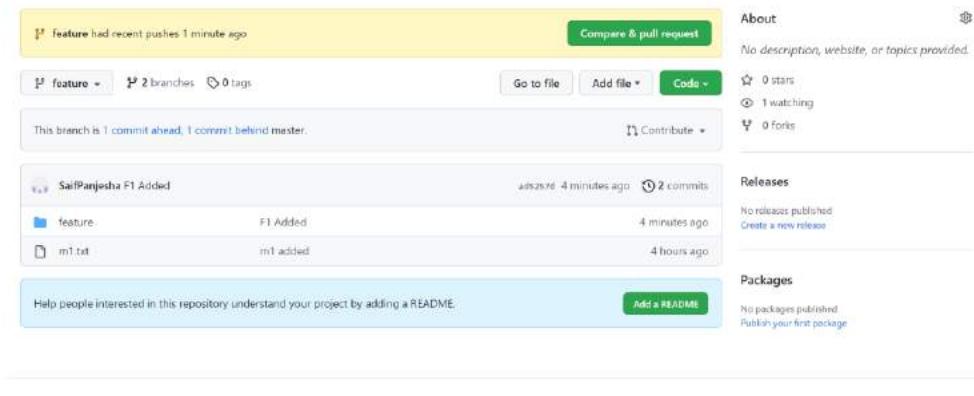


Fig. Successful added on GitHub

D:\one drive data\Desktop\gitHub1-basics>git branch -r

**origin/feature**

**origin/master**

D:\one drive data\Desktop\gitHub1-basics>git branch

\* feature

master

---

D:\one drive data\Desktop\gitHub1-basics>git branch -a

\* feature

master

remotes/origin/feature

remotes/origin/master

---

## Creating New Branch on GitHub

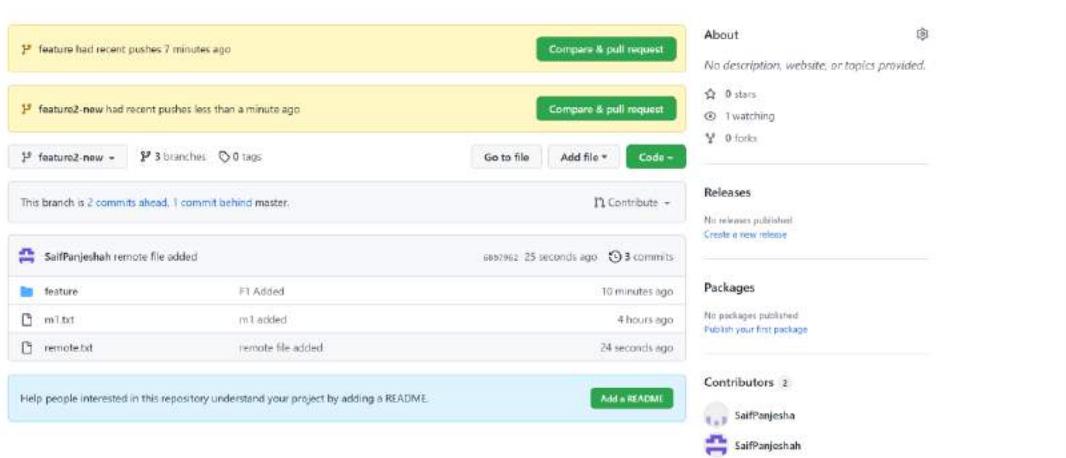


Fig. Created new Branch on Git Hub

The screenshot shows the VS Code interface with the 'GITHUB1-BASICS' repository open. The Explorer sidebar shows files m1.txt, f1.txt, feature, master, and f1.txt. The Terminal tab displays the command-line output of a git session:

```
D:\one drive data\Desktop\gitHub1-basics>git ls-remote
From https://github.com/SaifPanjeshah/gitHub1-basics.git
 13c36418a467824c7c0d92cf5883dc83277fb22c      HEAD
  ad5257df399e536b96e5b6c056c81d6435c232c0      refs/heads/feature
  68b796283e50d636a3de3e1d25859401ddc82a59      refs/heads/feature2-new
  13c36418a467824c7c0d92cf5883dc83277fb22c      refs/heads/master

D:\one drive data\Desktop\gitHub1-basics>git branch -a
* feature
  ls-rem
  master
  remotes/origin/feature
  remotes/origin/master

D:\one drive data\Desktop\gitHub1-basics>
```

Fig. Shows New Head Branch added in remote repository

## How Can View this remote Branch feature2-new in git (local repository)?

Git fetch retrieves the latest state.

Git fetch origin works because it updates the remote tracking branch.

The screenshot shows the VS Code interface with the 'GITHUB...' repository open. The Explorer sidebar shows files m1.txt, f1.txt, feature, master, and m1.txt. The Terminal tab displays the command-line output of a git session:

```
D:\one drive data\Desktop\gitHub1-basics>git fetch origin
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 689 bytes | 49.00 KiB/s, done.
From https://github.com/SaifPanjeshah/gitHub1-basics
 * [new branch]      feature2-new -> origin/feature2-new

D:\one drive data\Desktop\gitHub1-basics>git ls-remote
From https://github.com/SaifPanjeshah/gitHub1-basics.git
  13c36418a467824c7c0d92cf5883dc83277fb22c      HEAD
  ad5257df399e536b96e5b6c056c81d6435c232c0      refs/heads/feature
  68b796283e50d636a3de3e1d25859401ddc82a59      refs/heads/feature2-new
  13c36418a467824c7c0d92cf5883dc83277fb22c      refs/heads/master
```

Fig. Shows git fetch remote retrieves latest state

The screenshot shows a VS Code interface with the Terminal tab selected. The terminal window displays the following command-line session:

```

feature
* master
D:\one\drive\data\Desktop\gitHub1-basics>git branch -a
  feature
* master
  remotes/origin/feature
  remotes/origin/feature2-new
  remotes/origin/master

D:\one\drive\data\Desktop\gitHub1-basics>git checkout remotes/origin/feature2-new
Note: switching to 'remotes/origin/feature2-new'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

```

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 68b7962 remote file added

D:\one\drive\data\Desktop\gitHub1-basics>

Fig. Shows remote branch only retrieve latest change and in detached mode

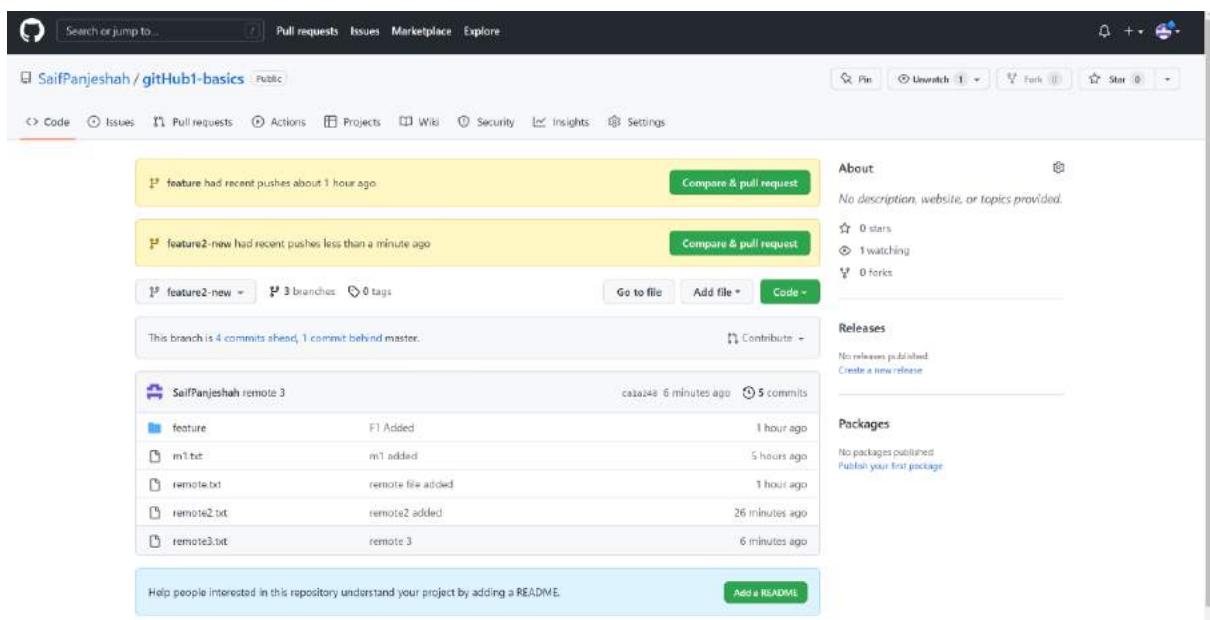


Fig. Shows Feature2 branch in GitHub

Now, to view complete changes we have to use git pull command

**git pull →** used to fetch and download content from a remote repository and immediately update the local repository to match that content(merge + fetch)



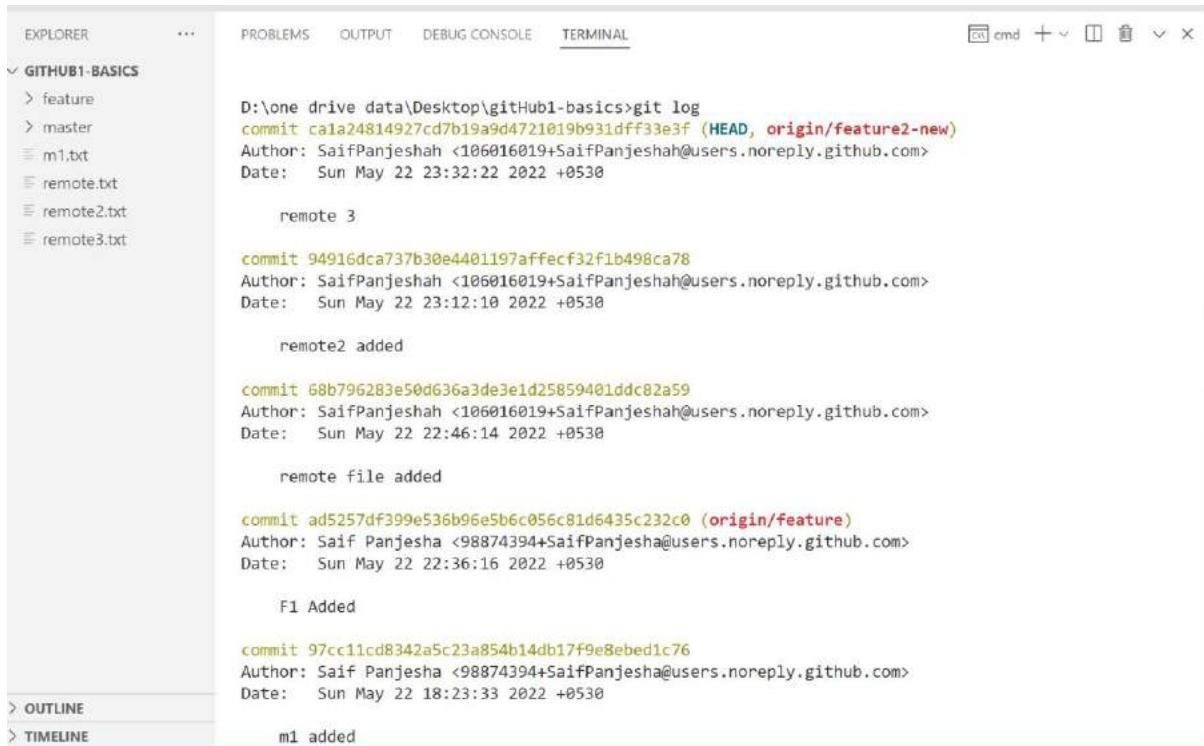
The screenshot shows the VS Code interface with the terminal tab selected. The terminal window displays the following command and its execution:

```
D:\one drive data\Desktop\gitHub1-basics>git branch
  feature
* master

D:\one drive data\Desktop\gitHub1-basics>git pull origin
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), 1.22 KB | 34.00 KB/s, done.
From https://github.com/SaifPanjeshah/gitHub1-basics
  68b7962..ca1a248  feature2-new -> origin/feature2-new
You asked to pull from the remote 'origin', but did not specify
a branch. Because this is not the default configured remote
for your current branch, you must specify a branch on the command line.

D:\one drive data\Desktop\gitHub1-basics>git
```

Fig. Shows git pull successful added to local repository



The screenshot shows the VS Code interface with the terminal tab selected. The terminal window displays the following command and its execution:

```
D:\one drive data\Desktop\gitHub1-basics>git log
commit ca1a24814927cd7b19a9d4721019b931dff33e3f (HEAD, origin/feature2-new)
Author: SaifPanjeshah <106016019+SaifPanjeshah@users.noreply.github.com>
Date:   Sun May 22 23:32:22 2022 +0530

    remote 3

commit 94916dca737b30e4401197affecf32f1b498ca78
Author: SaifPanjeshah <106016019+SaifPanjeshah@users.noreply.github.com>
Date:   Sun May 22 23:12:10 2022 +0530

    remote2 added

commit 68b796283e50d636a3de3e1d25859401ddc82a59
Author: SaifPanjeshah <106016019+SaifPanjeshah@users.noreply.github.com>
Date:   Sun May 22 22:46:14 2022 +0530

    remote file added

commit ad5257df399e536b96e5b6c056c81d6435c232c0 (origin/feature)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Sun May 22 22:36:16 2022 +0530

    F1 Added

commit 97cc11cd8342a5c23a854b14db17f9e8ebbed1c76
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Sun May 22 18:23:33 2022 +0530

    m1 added
```

Fig. Shows full commit history of GitHub feature2 -new Branch in local repository

## Understanding Local Tracking Branches:



Fig. Overview of Branches

A screenshot of a terminal window in a code editor interface. The window has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL, with TERMINAL selected. The terminal shows the following command-line session:

```
D:\one drive data\Desktop\GitHub1-basics>git remote
origin

D:\one drive data\Desktop\GitHub1-basics>git remote show origin
* remote origin
  Fetch URL: https://github.com/SaifPanjeshah/gitHub1-basics.git
  Push URL: https://github.com/SaifPanjeshah/gitHub1-basics.git
  HEAD branch: master
  Remote branches:
    feature      tracked
    feature2-new tracked
    master       tracked
  Local branch configured for 'git pull':
    feature2-new merges with remote feature2-new
  Local refs configured for 'git push':
    feature      pushes to feature      (fast-forwardable)
    feature2-new pushes to feature2-new (up to date)
    master       pushes to master     (up to date)

D:\one drive data\Desktop\GitHub1-basics>]
```

Fig. Commands on terminal

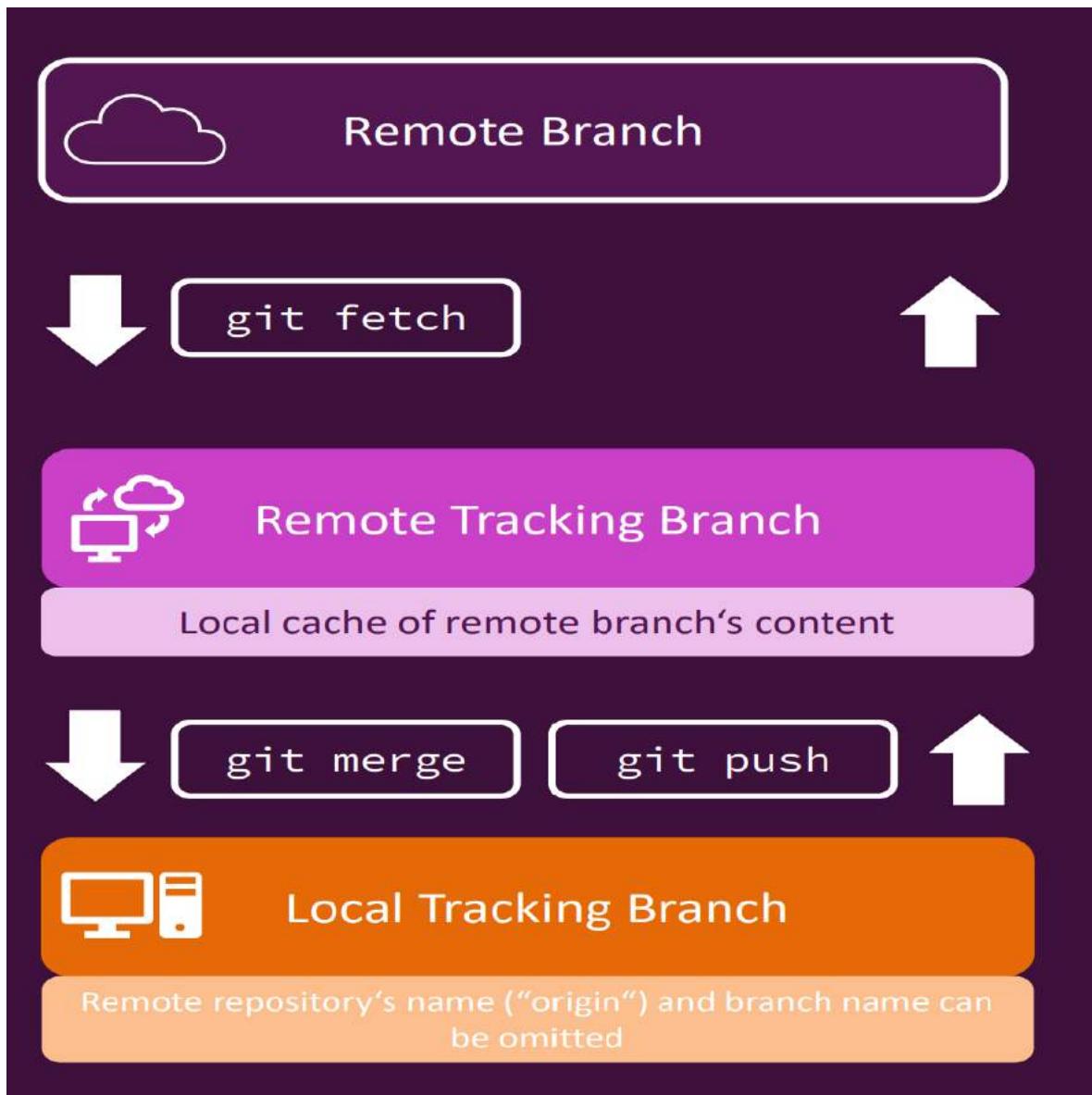
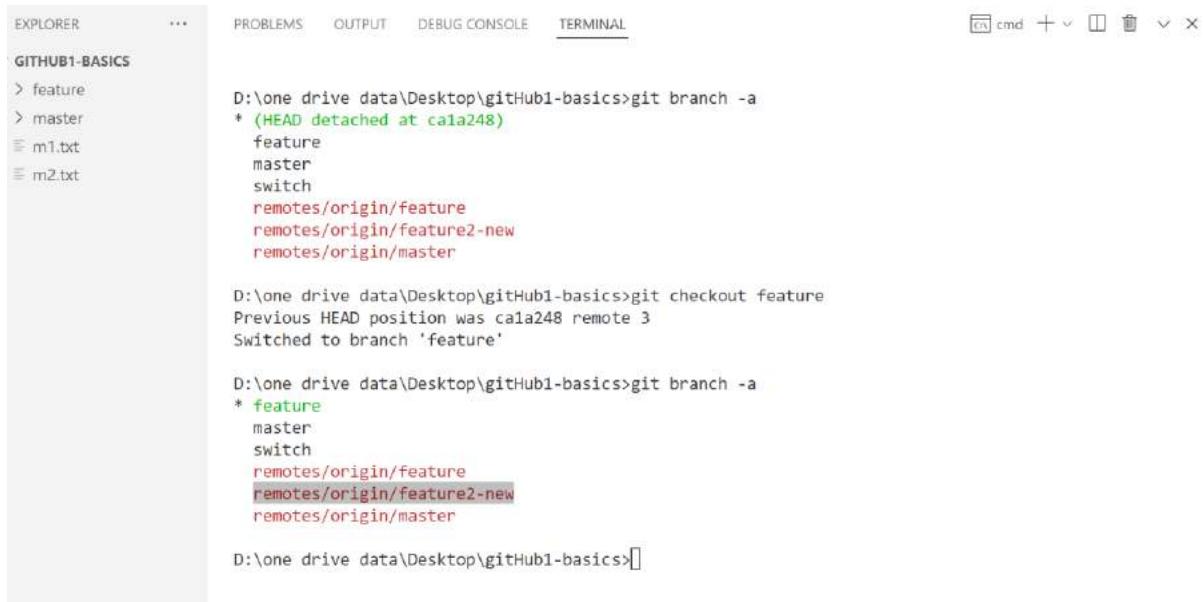


Fig. Local & remote tracking branches

## Creating Local Tracking Branches:



The screenshot shows a terminal window with the following content:

```
D:\one drive data\Desktop\gitHub1-basics>git branch -a
* (HEAD detached at ca1a248)
  feature
  master
  switch
  remotes/origin/feature
  remotes/origin/feature2-new
  remotes/origin/master

D:\one drive data\Desktop\gitHub1-basics>git checkout feature
Previous HEAD position was ca1a248 remote 3
Switched to branch 'feature'

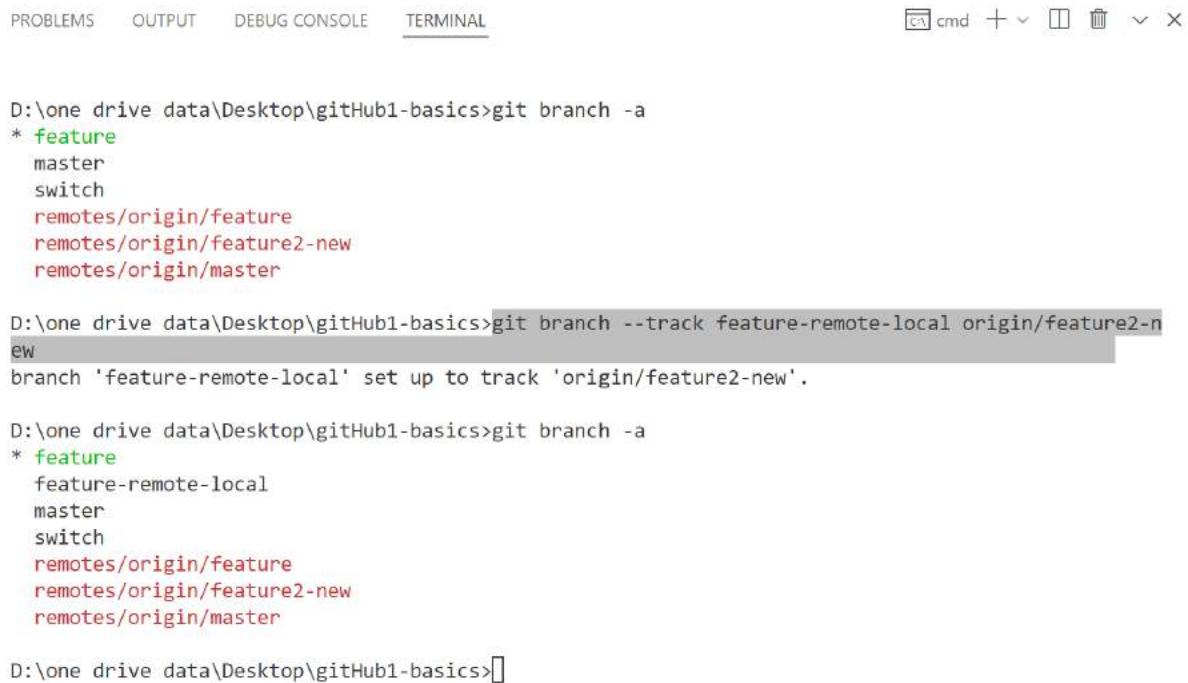
D:\one drive data\Desktop\gitHub1-basics>git branch -a
* feature
  master
  switch
  remotes/origin/feature
  remotes/origin/feature2-new
  remotes/origin/master

D:\one drive data\Desktop\gitHub1-basics>
```

Fig. checking out all branches

Now the goal, is to create local tracking branch of (remotes/origin/feature2-new) remote tracking branch:

**git branch --track feature-remote-local origin/feature2-new**



The screenshot shows a terminal window with the following content:

```
D:\one drive data\Desktop\gitHub1-basics>git branch -a
* feature
  master
  switch
  remotes/origin/feature
  remotes/origin/feature2-new
  remotes/origin/master

D:\one drive data\Desktop\gitHub1-basics>git branch --track feature-remote-local origin/feature2-new
branch 'feature-remote-local' set up to track 'origin/feature2-new'.

D:\one drive data\Desktop\gitHub1-basics>git branch -a
* feature
  feature-remote-local
  master
  switch
  remotes/origin/feature
  remotes/origin/feature2-new
  remotes/origin/master

D:\one drive data\Desktop\gitHub1-basics>
```

Fig. Local tracking Branches created as structured

The screenshot shows the VS Code interface with the Terminal tab selected. The terminal output is as follows:

```
D:\one drive data\Desktop\gitHub1-basics>git checkout feature-remote-local
Switched to branch 'feature-remote-local'
Your branch is up to date with 'origin/feature2-new'.

D:\one drive data\Desktop\gitHub1-basics>gi branch -a
'gi' is not recognized as an internal or external command,
operable program or batch file.

D:\one drive data\Desktop\gitHub1-basics>git branch -a
  feature
* feature-remote-local
  master
  switch
  remotes/origin/feature
  remotes/origin/feature2-new
  remotes/origin/master

D:\one drive data\Desktop\gitHub1-basics>git push
fatal: The upstream branch of your current branch does not match
the name of your current branch. To push to the upstream branch
on the remote, use

  git push origin HEAD:feature2-new

To push to the branch of the same name on the remote, use

  git push origin HEAD

To choose either option permanently, see push.default in 'git help config'.
```

Fig. after pushing the upstream branches not matching

**Now, to avoid this fatal error we have to create same branch as named in remote tracking branches**

The screenshot shows the VS Code interface with the Terminal tab selected. The terminal output is as follows:

```
D:\one drive data\Desktop\gitHub1-basics>git switch feature
Switched to branch 'feature'

D:\one drive data\Desktop\gitHub1-basics>git branch -a
* feature
  feature-remote-local
  master
  switch
  remotes/origin/feature
  remotes/origin/feature2-new
  remotes/origin/master

D:\one drive data\Desktop\gitHub1-basics>git branch -D feature-remote-local
Deleted branch feature-remote-local (was ca1a248).

D:\one drive data\Desktop\gitHub1-basics>git branch --track feature2-new origin/feature2-new
branch 'feature2-new' set up to track 'origin/feature2-new'.

D:\one drive data\Desktop\gitHub1-basics>git branch -a
* feature
  feature2-new
  master
  switch
  remotes/origin/feature
  remotes/origin/feature2-new
  remotes/origin/master

D:\one drive data\Desktop\gitHub1-basics>
```

Fig. created same name as remote tracking branch

The screenshot shows the VS Code interface with the Explorer, Terminal, and Output tabs visible. In the Explorer sidebar, under the 'GITHUB...' section, there is a 'remote tracking' folder containing 'remote.txt'. The terminal tab shows the following command-line session:

```
D:\one\drive\data\Desktop\gitHub1-basics>git branch -a
* feature
  feature2-new
  master
  switch
  remotes/origin/feature
  remotes/origin/feature2-new
  remotes/origin/master

D:\one\drive\data\Desktop\gitHub1-basics>git switch feature2-new
Switched to branch 'feature2-new'
Your branch is up to date with 'origin/feature2-new'.

D:\one\drive\data\Desktop\gitHub1-basics>git add .

D:\one\drive\data\Desktop\gitHub1-basics>git commit -m "remote added from local tracking branch"
[feature2-new e9e34c7] remote added from local tracking branch
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 remote tracking/remote.txt

D:\one\drive\data\Desktop\gitHub1-basics>
```

Fig. Added new file remote.txt

## Now push, on remote tracking branches:

The screenshot shows the VS Code interface with the Explorer, Terminal, and Output tabs visible. In the Explorer sidebar, under the 'GITHUB...' section, there is a 'remote tracking' folder containing 'remote.txt'. The terminal tab shows the following command-line session:

```
D:\one\drive\data\Desktop\gitHub1-basics>git push
fatal: credential-cache unavailable; no unix socket support
fatal: credential-cache unavailable; no unix socket support
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 357 bytes | 357.00 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.
To https://github.com/SaifPanjeshah/gitHub1-basics.git
  ca1a248..e9e34c7  feature2-new -> feature2-new

D:\one\drive\data\Desktop\gitHub1-basics>
```

Fig. Success on remote tracking branches

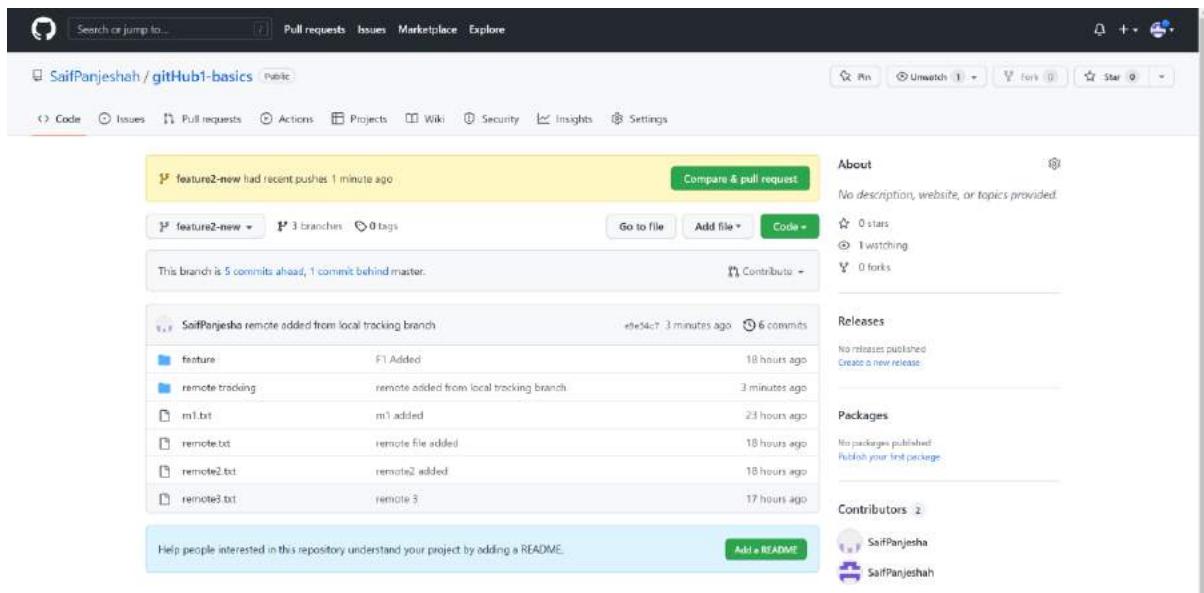


Fig. Successful on remote added from local tracking branches

**Now let's try for pull from remote tracking branches to local tracking branches**

**Created New File :**

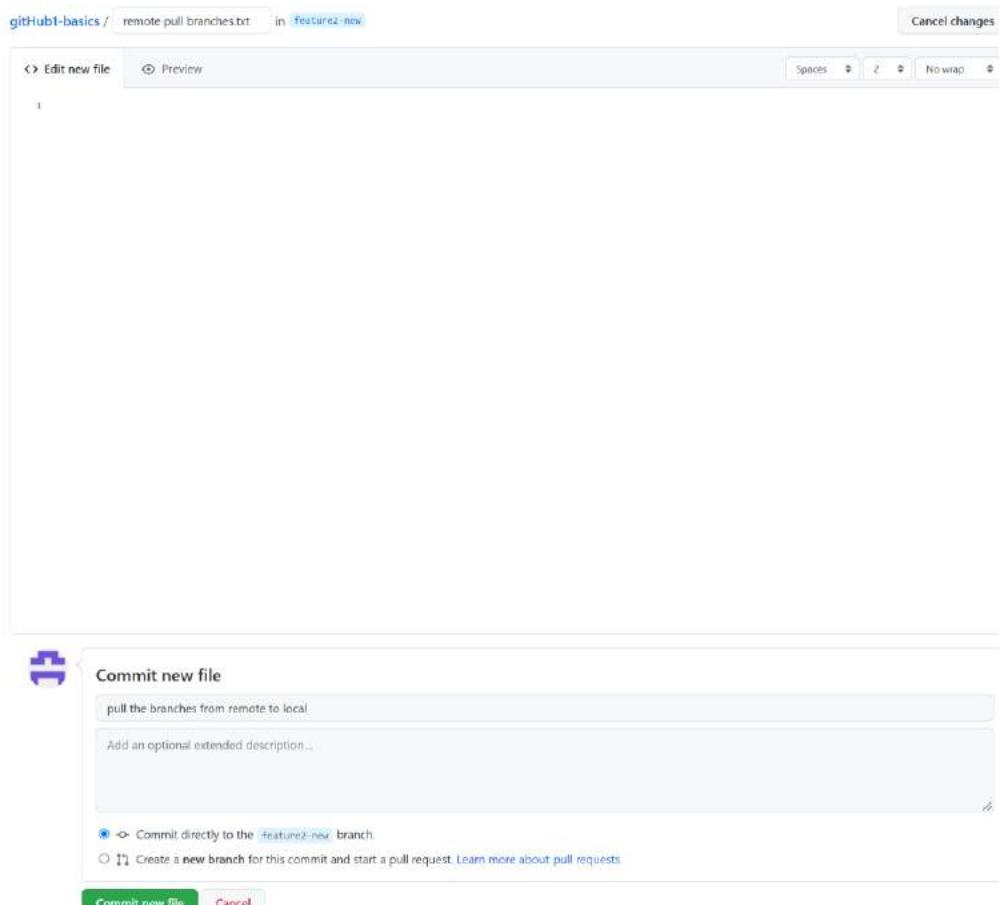


Fig. Created a new File on remote tracking branches

## git branch -vv: List local tracking branches and their remotes

The screenshot shows the VS Code interface with the terminal tab active. The terminal window displays the following command-line session:

```
D:\One Drive\data\Desktop\GitHub1-Basics>git pull
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 2 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (2/2), 651 bytes | 108.00 KiB/s, done.
From https://github.com/SaifPanjeshah/GitHub1-Basics
  e9e34c7..9e3af21  feature2-new -> origin/feature2-new
Updating e9e34c7..9e3af21
Fast-forward
 remote pull branches.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 remote pull branches.txt

D:\One Drive\data\Desktop\GitHub1-Basics>git branch -vv
  feature      9457021 Merge branch 'master' of https://github.com/SaifPanjeshah/GitHub1-Basics into feature
* feature2-new 9e3af21 [origin/feature2-new] pull the branches from remote to local
  master       13c3641 m2 added
  switch       13c3641 m2 added

D:\One Drive\data\Desktop\GitHub1-Basics>
```

Fig. Successful added to local tracking branches from remote vice versa

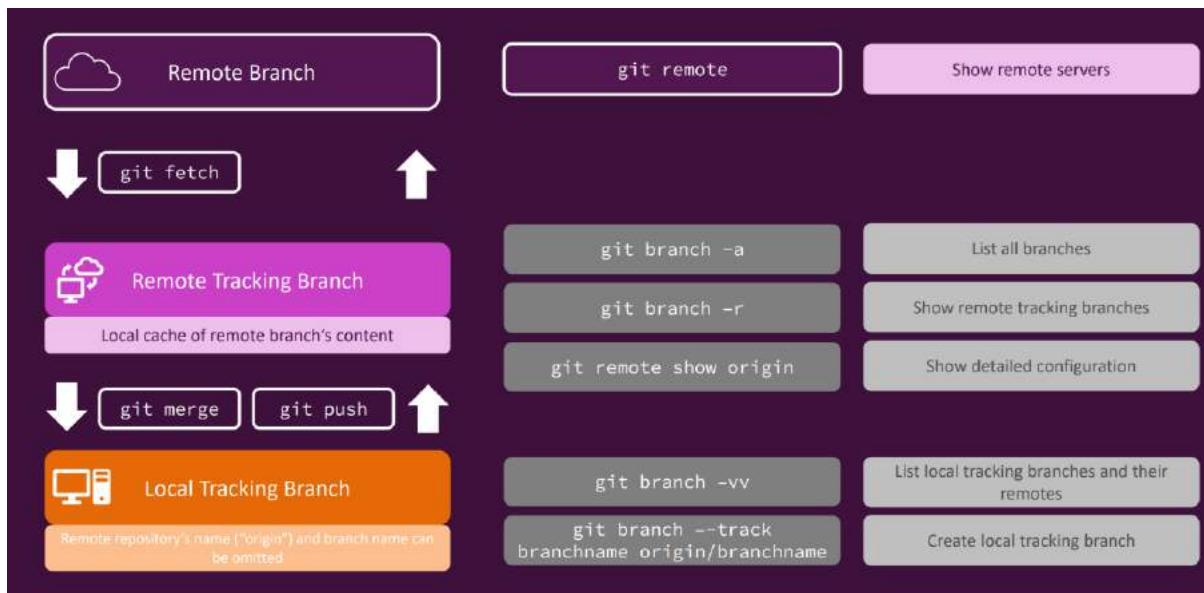


Fig. Local & Remote Tracking Branches Commands

## Cloning a remote repository:

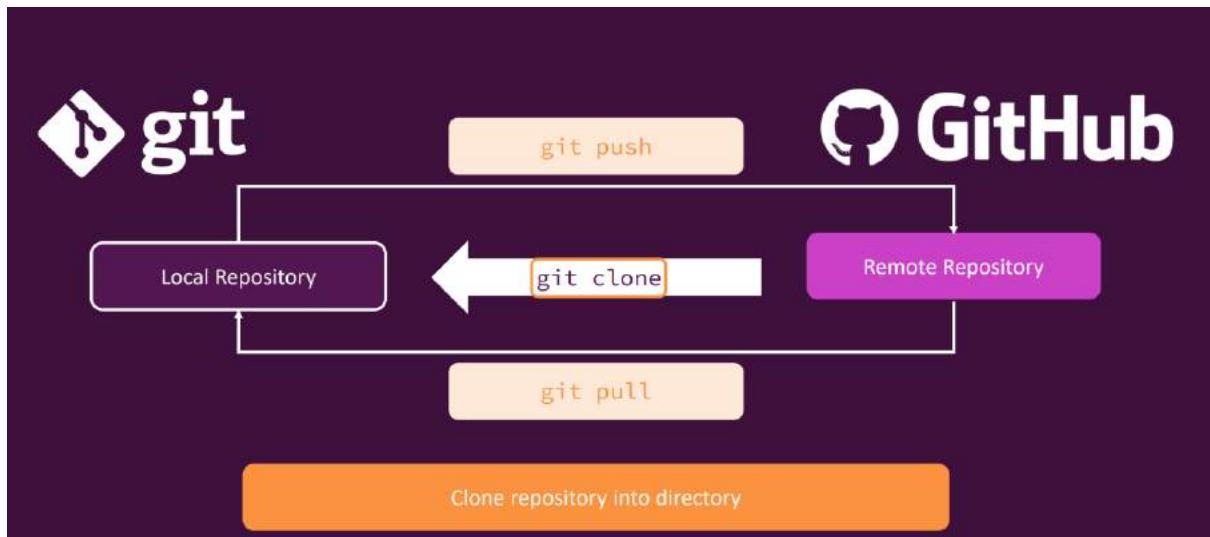


Fig. Shows How to Clone a remote repository

**git clone** is a git command line utility used to target an existing repository and create a clone, or copy of the target repository.

A screenshot of the Visual Studio Code interface. The left sidebar shows an "EXPLORER" view with a "CLONE" section containing a folder named "gitHub1-basics". The main workspace shows a large "X" icon, indicating a pending operation. The bottom status bar shows the path "D:\One Drive Data\Desktop\clone". The terminal tab is active, displaying the output of the "git clone" command:

```
D:\One Drive Data\Desktop\clone>git clone https://github.com/SaifPanjeshah/gitHub1-basics.git
Cloning into 'gitHub1-basics'...
remote: Enumerating objects: 20, done.
remote: Counting objects: 100% (20/20), done.
remote: Compressing objects: 100% (13/13), done.
remote: Total 20 (delta 7), reused 9 (delta 2), pack-reused 0
Receiving objects: 100% (20/20), done.
Resolving deltas: 100% (7/7), done.

D:\One Drive Data\Desktop\clone>[]
```

Fig. Cloning repository into folder

```
D:\One Drive Data\Desktop\clone>git status
fatal: not a git repository (or any of the parent directories): .git
```

Fig. Fatal error while using git commands

## How to resolve this fatal error?

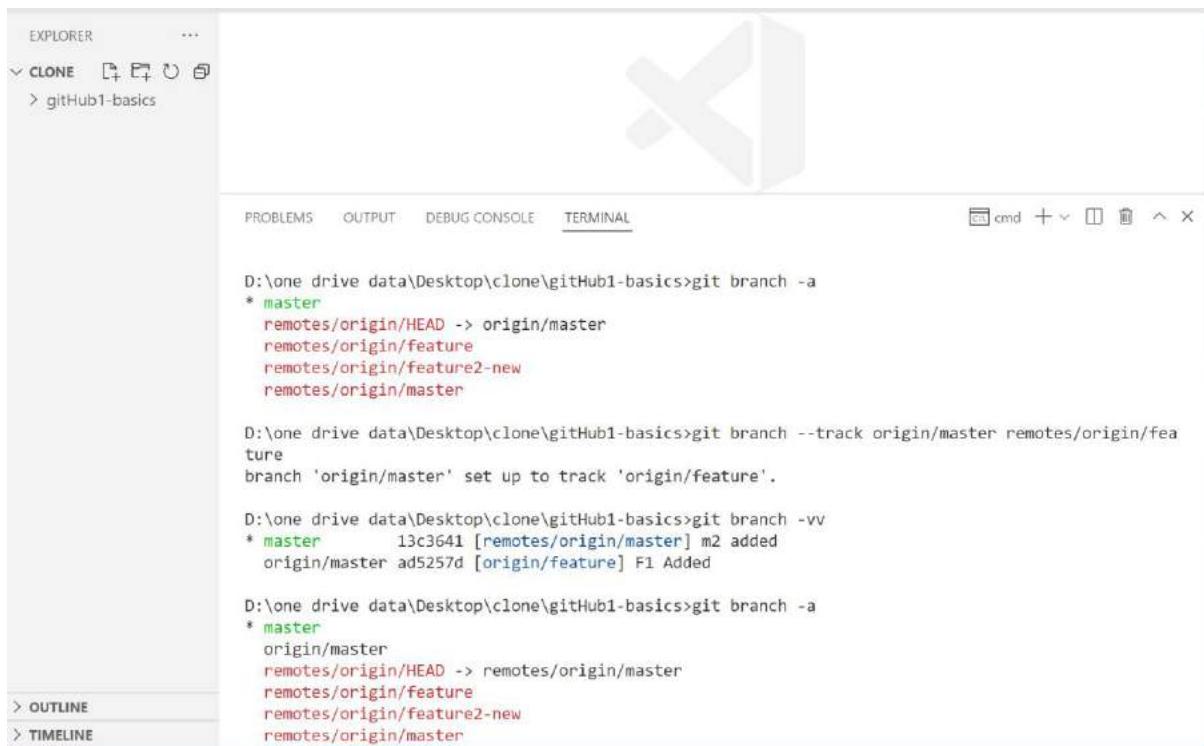


The screenshot shows the VS Code interface. In the Explorer sidebar, there is a 'CLONE' section with a folder icon and the path 'gitHub1-basics'. The terminal tab is active, displaying the following command-line session:

```
D:\one drive data\Desktop\clone>cd gitHub1-basics
D:\one drive data\Desktop\clone\gitHub1-basics>git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
D:\one drive data\Desktop\clone\gitHub1-basics>[]
```

Fig. resolve fatal error by changing directory to repository



The screenshot shows the VS Code interface. In the Explorer sidebar, there is a 'CLONE' section with a folder icon and the path 'gitHub1-basics'. The terminal tab is active, displaying the following command-line session:

```
D:\one drive data\Desktop\clone\gitHub1-basics>git branch -a
* master
  remotes/origin/HEAD -> origin/master
  remotes/origin/feature
  remotes/origin/feature2-new
  remotes/origin/master

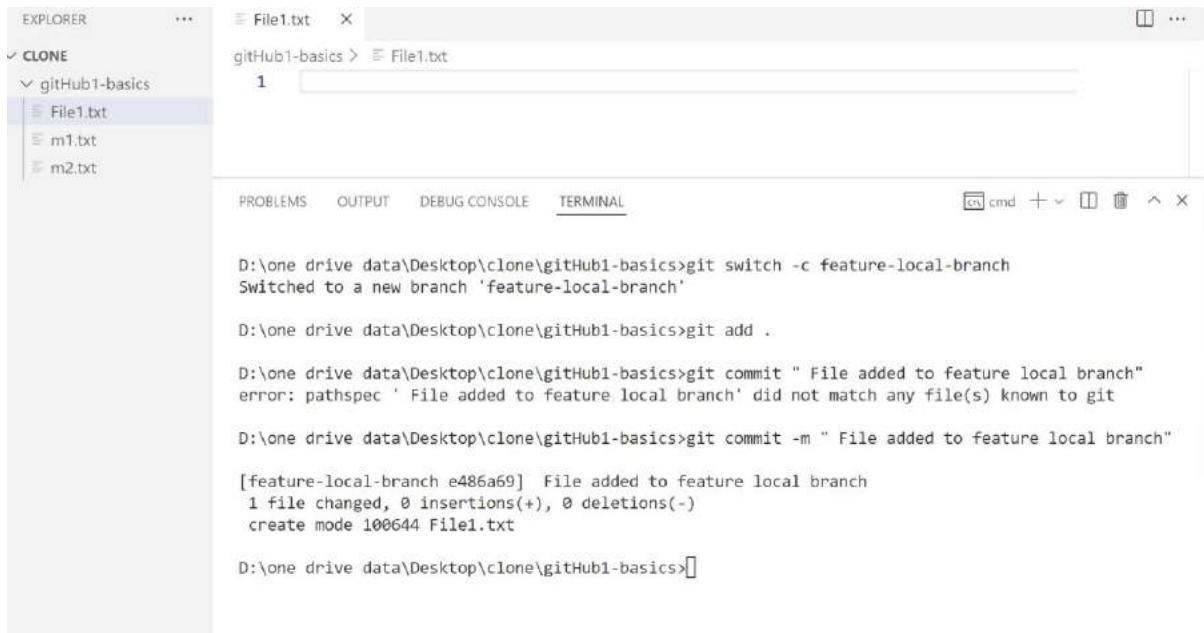
D:\one drive data\Desktop\clone\gitHub1-basics>git branch --track origin/master remotes/origin/fea
ture
branch 'origin/master' set up to track 'origin/feature'.

D:\one drive data\Desktop\clone\gitHub1-basics>git branch -vv
* master 13c3641 [remotes/origin/master] m2 added
          origin/master ad5257d [origin/feature] F1 Added

D:\one drive data\Desktop\clone\gitHub1-basics>git branch -a
* master
  origin/master
  remotes/origin/HEAD -> remotes/origin/master
  remotes/origin/feature
  remotes/origin/feature2-new
  remotes/origin/master
```

Fig. Changing head to feature branch

## Let's Create a New Branch in our local branch tracking:



The screenshot shows the VS Code interface with the Terminal tab selected. The Explorer sidebar shows a cloned repository named 'gitHub1-basics' containing files 'File1.txt', 'm1.txt', and 'm2.txt'. The terminal output shows the following commands and their results:

```
D:\one drive data\Desktop\clone\gitHub1-basics>git switch -c feature-local-branch
Switched to a new branch 'feature-local-branch'

D:\one drive data\Desktop\clone\gitHub1-basics>git add .

D:\one drive data\Desktop\clone\gitHub1-basics>git commit " File added to Feature local branch"
error: pathspec ' File added to feature local branch' did not match any file(s) known to git

D:\one drive data\Desktop\clone\gitHub1-basics>git commit -m " File added to feature local branch"

[feature-local-branch e486a69] File added to feature local branch
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 File1.txt

D:\one drive data\Desktop\clone\gitHub1-basics>
```

Fig. Created feature-local-branch



The screenshot shows the VS Code interface with the Terminal tab selected. The Explorer sidebar shows the same cloned repository 'gitHub1-basics' with files 'File1.txt', 'm1.txt', and 'm2.txt'. The terminal output shows the following commands and their results:

```
D:\one drive data\Desktop\clone\gitHub1-basics>git branch
* feature-local-branch
  master
  origin/master

D:\one drive data\Desktop\clone\gitHub1-basics>git push origin feature-local-branch
fatal: credential-cache unavailable; no unix socket support
fatal: credential-cache unavailable; no unix socket support
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 282 bytes | 282.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'feature-local-branch' on GitHub by visiting:
remote:     https://github.com/SaifPanjeshah/gitHub1-basics/pull/new/feature-local-branch
remote:
To https://github.com/SaifPanjeshah/gitHub1-basics.git
 * [new branch]      feature-local-branch -> feature-local-branch

D:\one drive data\Desktop\clone\gitHub1-basics>
```

Fig. Successful push on remote branch

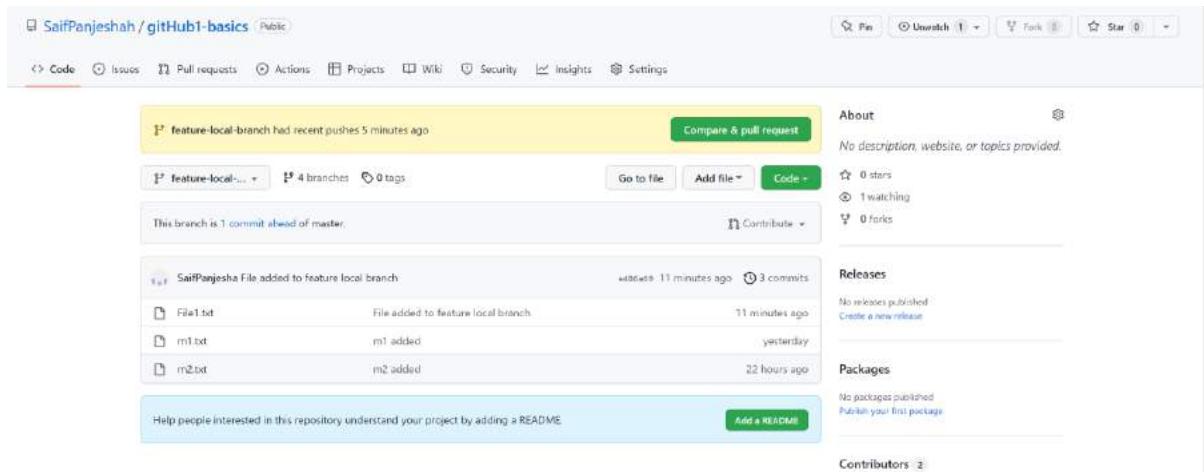


Fig. Successful added on remote branch GitHub

The screenshot shows a VS Code interface with the terminal tab active. The terminal window displays two sets of git branch commands. The first set shows all tracking branches with 'git branch -a', listing 'feature-local-branch', 'master', 'origin/master', 'remotes/origin/HEAD', 'remotes/origin/feature', 'remotes/origin/feature-local-branch', 'remotes/origin/feature2-new', and 'remotes/origin/master'. The second set shows detailed information with 'git branch -vv', showing 'feature-local-branch' with commit 'e486a69' and file 'm2.txt' added, and 'origin/master' with commit 'ad5257d' and file 'F1' added. The current directory is 'D:\one drive data\Desktop\clone\gitHub1-basics'.

```
D:\one drive data\Desktop\clone\gitHub1-basics>git branch -a
* feature-local-branch
  master
  origin/master
  remotes/origin/HEAD -> remotes/origin/master
  remotes/origin/feature
  remotes/origin/feature-local-branch
  remotes/origin/feature2-new
  remotes/origin/master

D:\one drive data\Desktop\clone\gitHub1-basics>git branch -vv
* feature-local-branch e486a69 File added to feature local branch
  master      13c3641 [remotes/origin/master] m2 added
  origin/master ad5257d [origin/feature] F1 Added

D:\one drive data\Desktop\clone\gitHub1-basics>
```

Fig. Shows all tracking branches

**Now, our feature-local-branch refer to any kind of well remote tracking branch? So How Can we turn into local tracking branch?**

```
D:\one drive data\Desktop\clone\gitHub1-basics>git branch -vv
* feature-local-branch e486a69 File added to feature local branch
  master      13c3641 [remotes/origin/master] m2 added
  origin/master ad5257d [origin/feature] F1 Added
```

```
D:\one drive data\Desktop\clone\gitHub1-basics> git switch master
Switched to branch 'master'
Your branch is up to date with 'remotes/origin/master'.
```

```
D:\one drive data\Desktop\clone\gitHub1-basics>git branch -D feature-local-branch
Deleted branch feature-local-branch (was e486a69).
```

```
D:\one drive data\Desktop\clone\gitHub1-basics>
```

The screenshot shows the VS Code interface with the Terminal tab selected. In the Explorer sidebar, there is a 'CLONE' folder containing a 'gitHub1-basics' repository with files 'm1.txt' and 'm2.txt'. The terminal window displays the following command history:

```
D:\one drive data\Desktop\clone\gitHub1-basics>git branch -a
* master
  origin/master
  remotes/origin/HEAD -> remotes/origin/master
  remotes/origin/feature
  remotes/origin/feature-local-branch
  remotes/origin/feature2-new
  remotes/origin/master

D:\one drive data\Desktop\clone\gitHub1-basics>git branch --track feature-local-branch origin/feature-local-branch
branch 'feature-local-branch' set up to track 'origin/feature-local-branch'.

D:\one drive data\Desktop\clone\gitHub1-basics>git branch -vv
  feature-local-branch e486a69 [origin/feature-local-branch] File added to feature local branch
* master           13c3641 [remotes/origin/master] m2 added
  origin/master     ad5257d [origin/feature] F1 Added

D:\one drive data\Desktop\clone\gitHub1-basics>[]
```

Fig. Created Local tracking branch

```
D:\one drive data\Desktop\clone\gitHub1-basics>git branch -a
  feature-local-branch
* master
  origin/master
  remotes/origin/HEAD -> remotes/origin/master
  remotes/origin/feature
  remotes/origin/feature-local-branch
  remotes/origin/feature2-new
  remotes/origin/master

D:\one drive data\Desktop\clone\gitHub1-basics>git switch  feature-local-branch
Switched to branch 'feature-local-branch'
Your branch is up to date with 'origin/feature-local-branch'.
```

```
D:\one drive data\Desktop\clone\gitHub1-basics>[]
```

Fig. Success Local tracking branch

## Understanding the Upstream:

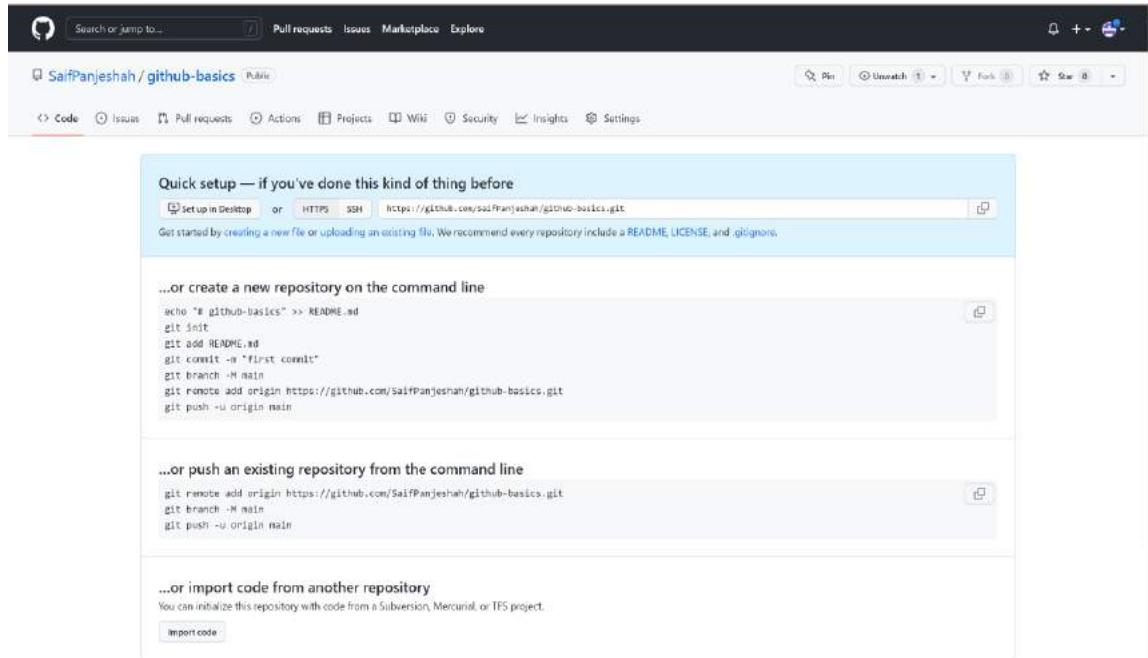


Fig. Checkout for “-u” Upstream

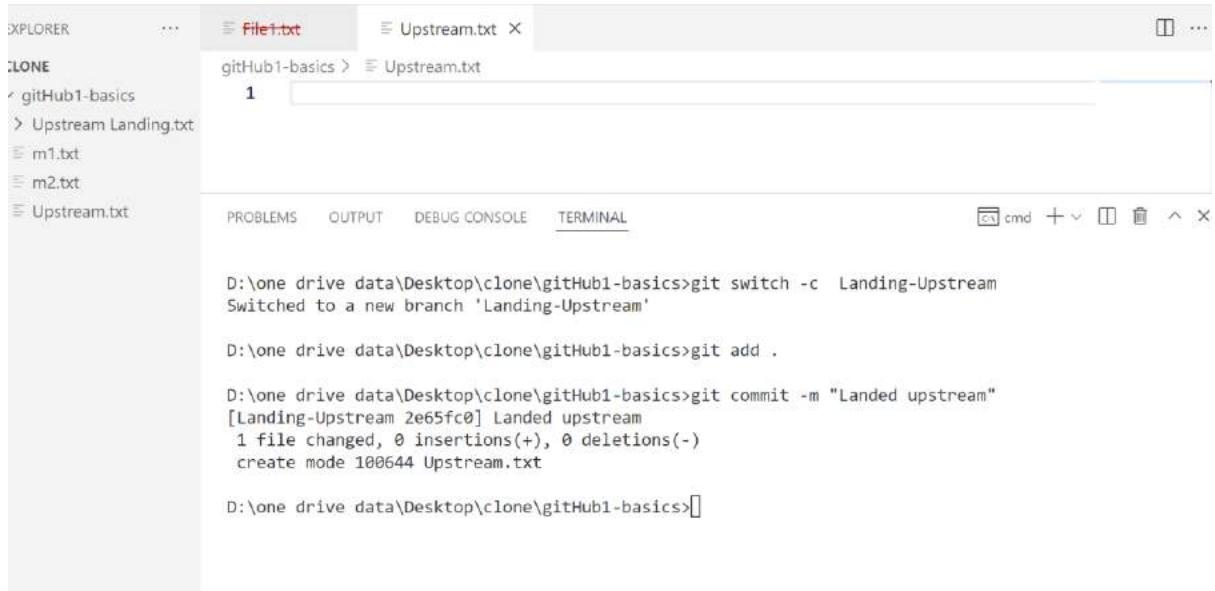


Fig. Created upstream branch

The screenshot shows a Microsoft Visual Studio Code interface with the 'TERMINAL' tab selected. The terminal window displays the following command-line session:

```
D:\one drive data\Desktop\clone\gitHub1-basics>git push -u origin Landing-Upstream
fatal: credential-cache unavailable; no unix socket support
fatal: credential-cache unavailable; no unix socket support
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 274 bytes | 274.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'Landing-Upstream' on GitHub by visiting:
remote:     https://github.com/SaifPanjeshah/gitHub1-basics/pull/new/Landing-Upstream
remote:
To https://github.com/SaifPanjeshah/gitHub1-basics.git
 * [new branch]      Landing-Upstream -> Landing-Upstream
branch 'Landing-Upstream' set up to track 'origin/Landing-Upstream'.

D:\one drive data\Desktop\clone\gitHub1-basics>git branch -vv
* Landing-Upstream 2e65fc0 [origin/Landing-Upstream] Landed upstream
  feature-local-branch e486a69 [origin/feature-local-branch] File added to feature local branch
  master              13c3641 [remotes/origin/master] m2 added
  origin/master       ad5257d [origin/feature] F1 Added

D:\one drive data\Desktop\clone\gitHub1-basics>
```

Fig. Successful added on remote without tracking the branches

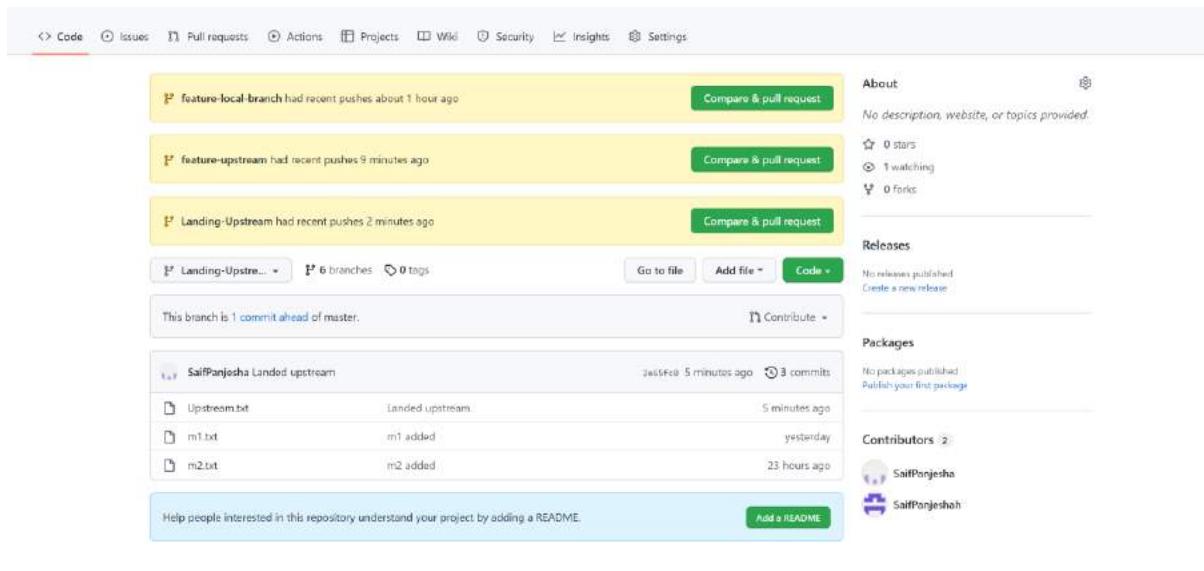
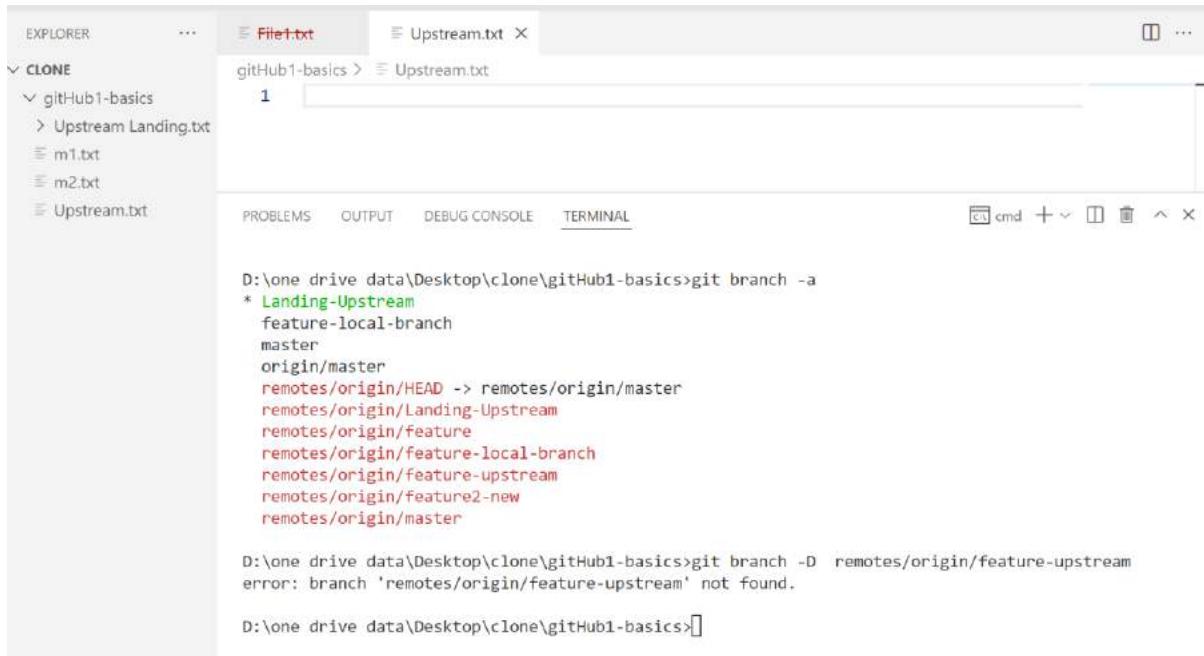


Fig. Successful Upstream.txt on GitHub

## Deleting Remote Branches & Public Commits:



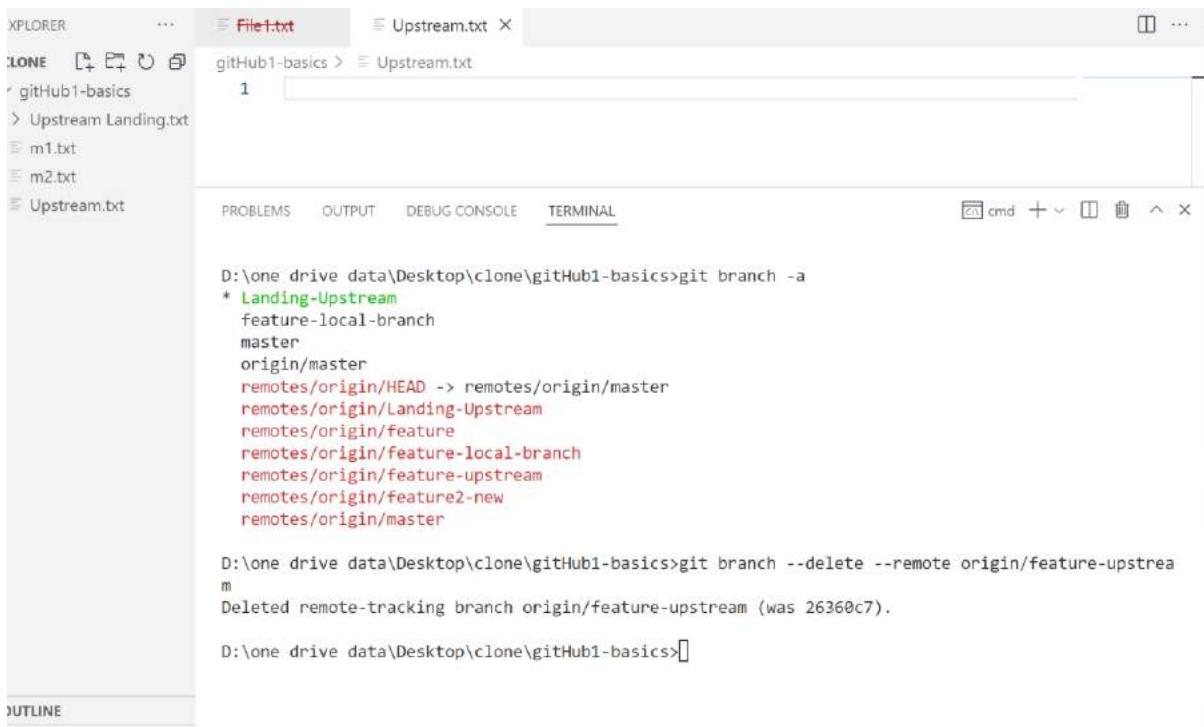
The screenshot shows the VS Code interface with the terminal tab active. The terminal output shows the following command and its result:

```
D:\one drive data\Desktop\clone\gitHub1-basics>git branch -D remotes/origin/feature-upstream
error: branch 'remotes/origin/feature-upstream' not found.
```

Fig. Not Working deletion

**How can we delete now?**

**git branch --delete --remote origin/feature-upstream**



The screenshot shows the VS Code interface with the terminal tab active. The terminal output shows the following command and its result:

```
D:\one drive data\Desktop\clone\gitHub1-basics>git branch -a
* Landing-Upstream
  feature-local-branch
  master
  origin/master
  remotes/origin/HEAD -> remotes/origin/master
  remotes/origin/Landing-Upstream
  remotes/origin/feature
  remotes/origin/feature-local-branch
  remotes/origin/feature-upstream
  remotes/origin/feature2-new
  remotes/origin/master

D:\one drive data\Desktop\clone\gitHub1-basics>git branch --delete --remote origin/feature-upstream
Deleted remote-tracking branch origin/feature-upstream (was 26360c7).

D:\one drive data\Desktop\clone\gitHub1-basics>
```

Fig. Deleted branches

The screenshot shows the VS Code interface with the terminal tab selected. The command 'git ls-remote' is run against the remote repository 'https://github.com/SaifPanjeshah/gitHub1-basics'. The output lists several branches and their corresponding commit hashes:

```
D:\one\drive\data\Desktop\clone\gitHub1-basics>git ls-remote
From https://github.com/SaifPanjeshah/gitHub1-basics.git
13c36418a467824c7c0d92cf5883dc83277fb22c      HEAD
2e65fc0c1f73d1287144608bf2f0b707e9ebc2d0      refs/heads/Landing-Upstream
ad5257df399e536b96e5b6c056c81d6435c232c0      refs/heads/feature
e486a694dbb2ac00a54696adedf1a2bc3c29b722      refs/heads/feature-local-branch
26360c7c2beec0726b027c0f3aed304738aa6b2d      refs/heads/feature-upstream
9e3af2165cc5f29e149eeea1207a96d2db77004b      refs/heads/feature2-new
13c36418a467824c7c0d92cf5883dc83277fb22c      refs/heads/master
```

Fig. ls-remotes shows that feature-upstream is in remote

## How can we delete now?

```
D:\one\drive\data\Desktop\clone\gitHub1-basics>git push origin --delete feature-upstream
fatal: credential-cache unavailable; no unix socket support
fatal: credential-cache unavailable; no unix socket support
To https://github.com/SaifPanjeshah/gitHub1-basics.git
 - [deleted]          feature-upstream

D:\one\drive\data\Desktop\clone\gitHub1-basics>git ls-remote
From https://github.com/SaifPanjeshah/gitHub1-basics.git
13c36418a467824c7c0d92cf5883dc83277fb22c      HEAD
2e65fc0c1f73d1287144608bf2f0b707e9ebc2d0      refs/heads/Landing-Upstream
ad5257df399e536b96e5b6c056c81d6435c232c0      refs/heads/feature
e486a694dbb2ac00a54696adedf1a2bc3c29b722      refs/heads/feature-local-branch
26360c7c2beec0726b027c0f3aed304738aa6b2d      refs/heads/feature-upstream
9e3af2165cc5f29e149eeea1207a96d2db77004b      refs/heads/feature2-new
13c36418a467824c7c0d92cf5883dc83277fb22c      refs/heads/master
```

Fig. Successful deleted

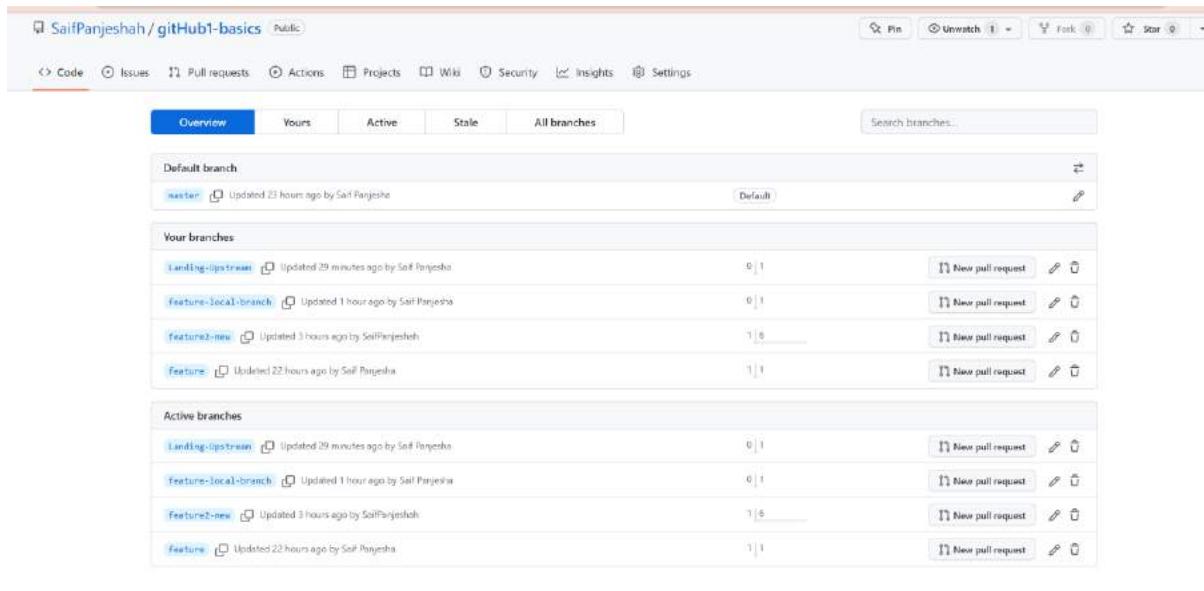
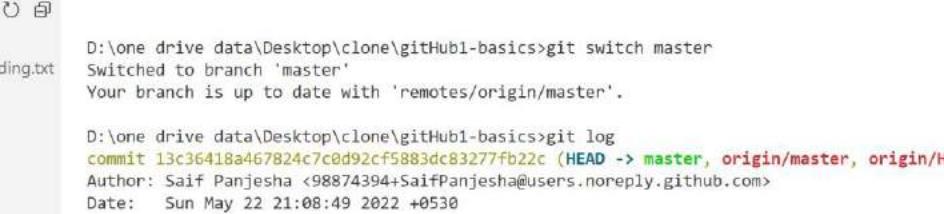


Fig. feature-upstream successful deleted on GitHub

## **How to delete Public Commits:**



The screenshot shows a Visual Studio Code interface with the following details:

- Explorer View:** Shows a file tree with files: `gitHub1-basics`, `Upstream Landing.txt`, `m1.txt`, and `m2.txt`.
- Terminal Tab:** Active tab, showing a command-line session in a Windows terminal window.
- Terminal Output:**

```
D:\one\drive\data\Desktop\clone\gitHub1-basics>git switch master
Switched to branch 'master'
Your branch is up to date with 'remotes/origin/master'.

D:\one\drive\data\Desktop\clone\gitHub1-basics>git log
commit 13c36418a467824c7c0d92cf5883dc83277fb22c (HEAD -> master, origin/master, origin/HEAD)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Sun May 22 21:08:49 2022 +0530

    m2 added

commit 97cc11cd8342a5c23a854b14db17f9e8ebbed1c76
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Sun May 22 18:23:33 2022 +0530

    m1 added

D:\one\drive\data\Desktop\clone\gitHub1-basics>
```

## Fig. Public Commits

```

D:\one drive data\Desktop\clone\gitHub1-basics>git log
commit 13c36418a467824c7c0d92cf5883dc83277fb22c (HEAD -> master, origin/master, origin/HEAD)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Sun May 22 21:08:49 2022 +0530

    m2 added

commit 97cc11cd8342a5c23a854b14db17f9e8ebcd1c76
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Sun May 22 18:23:33 2022 +0530

    m1 added

D:\one drive data\Desktop\clone\gitHub1-basics>git reset --hard HEAD~1
HEAD is now at 97cc11c m1 added

D:\one drive data\Desktop\clone\gitHub1-basics>git push origin master
fatal: credential-cache unavailable; no unix socket support
fatal: credential-cache unavailable; no unix socket support
To https://github.com/SaifPanjeshah/gitHub1-basics.git
! [rejected]      master -> master (non-fast-forward)
error: failed to push some refs to 'https://github.com/SaifPanjeshah/gitHub1-basics.git'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Integrate the remote changes (e.g.
hint: 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

D:\one drive data\Desktop\clone\gitHub1-basics>[]

```

Fig. Public Commits deleted unable to push

## How can we delete this commit?

```

D:\one drive data\Desktop\clone\gitHub1-basics>git push --force origin master
fatal: credential-cache unavailable; no unix socket support
fatal: credential-cache unavailable; no unix socket support
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/SaifPanjeshah/gitHub1-basics.git
 + 13c3641...97cc11c master -> master (forced update)

D:\one drive data\Desktop\clone\gitHub1-basics>git pull origin master
From https://github.com/SaifPanjeshah/gitHub1-basics
 * branch            master      -> FETCH_HEAD
Already up to date.

D:\one drive data\Desktop\clone\gitHub1-basics>[]

```

Fig. Successful push deleted commits and pull master

The screenshot shows a GitHub repository page for 'SaifPanjeshah / gitHub1-basics'. The repository is public. At the top, there are navigation links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. A yellow banner at the top indicates that 'Landing-Upstream' had recent pushes 36 minutes ago. Below the banner, it shows the master branch (5 branches, 0 tags), a commit by 'SaifPanjeshah' adding 'm1' (commit hash: 97cc11c, yesterday, 1 commit), and a file 'm1.txt' added. On the right side, there's an 'About' section with 'No description', 0 stars, 1 watching, 0 forks, and a 'Releases' section with 'No releases published' and 'Create a new release'. A button to 'Add a README' is also present.

Fig. Success on Deletion GitHub

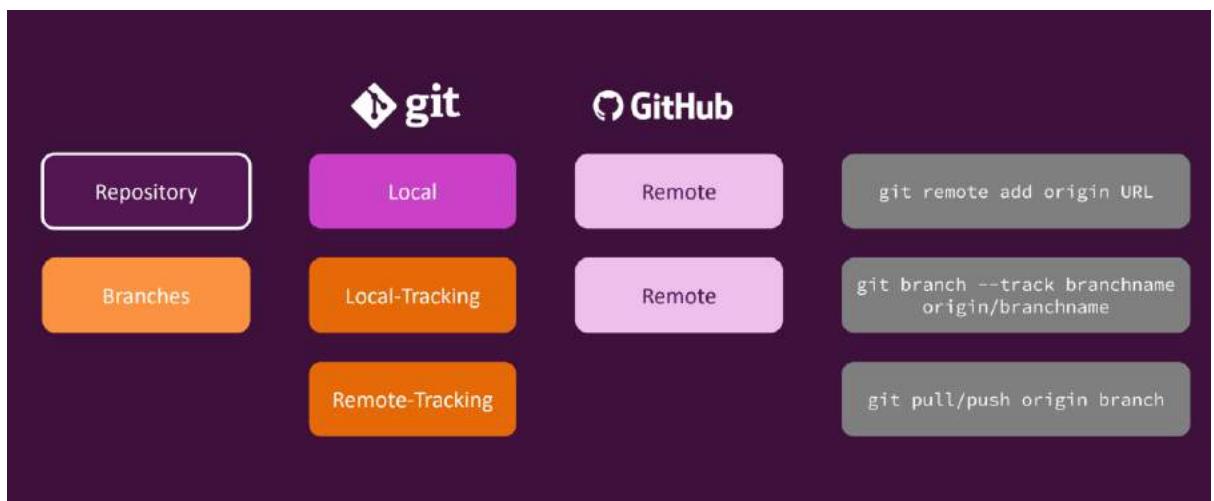


Fig. Wrap up summary

## Useful Resources & Links

GitHub official website => <https://github.com/>

GitHub pricing => <https://github.com/pricing>

## **GitHub Deep Dive – Collaboration & Contribution**

### **Module Introduction:**



Fig. Module Introduction

### **Module Content:**

- 1. Understand GitHub Accounts & Repository Types**
- 2. Collaborating to GitHub & Contributing to open-Source Project**
- 3. Creating your GitHub Portfolio Page & More Features to Explore**

## Why we use Git Hub:

### Four Core Reasons Why GitHub:

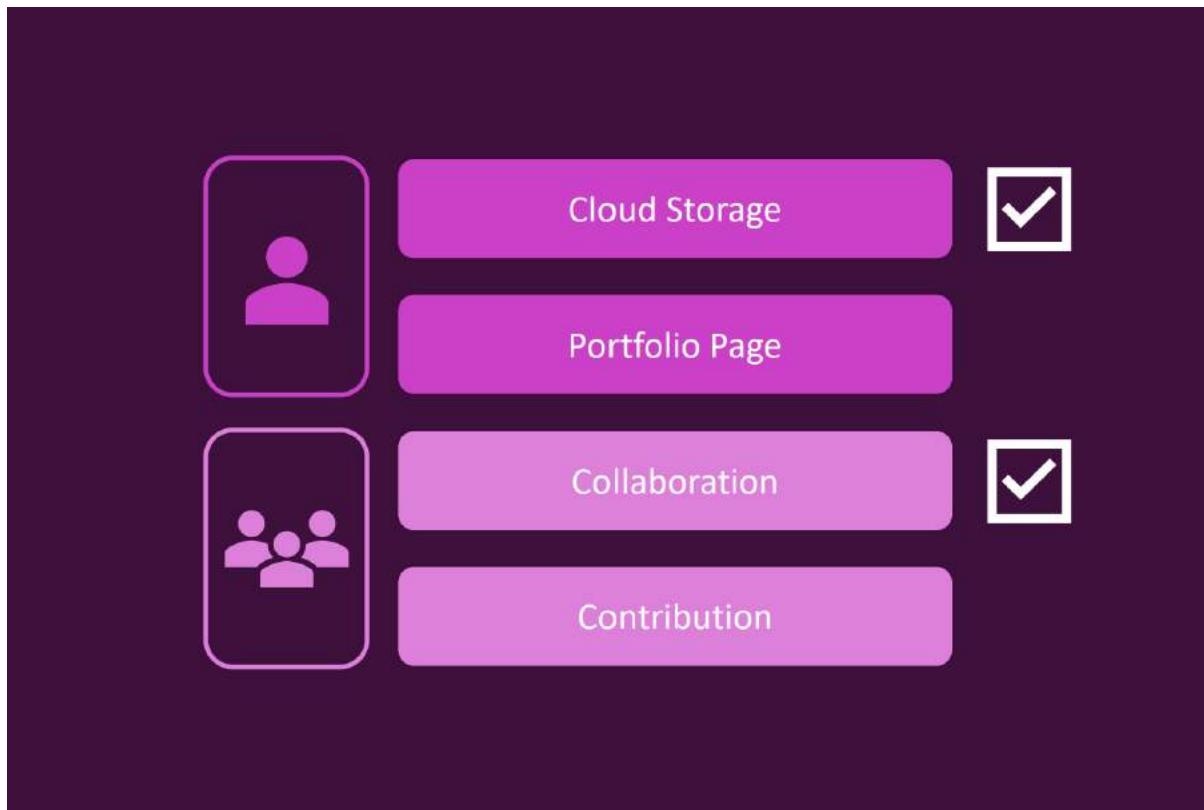


Fig. The 4 GitHub Use Cases

- **Cloud Storage:** the single user uses GitHub, for example, to have a cloud storage of his or her own Git projects.
- **Portfolio Page:** The single user might also use GitHub to present a portfolio page, so a page presenting all the core skills, all the great projects he or she worked on, or is working on.
- **Collaboration:** Collaboration GitHub means that either you have a project, but you want to add other people, but to collaborate with you on this project
- **Contribution:** It builds your resume by demonstrating that you can collaborate with others on code.

## Understanding The Account Types:



Fig. Different Accounts in GitHub

Pricing Models official site: <https://github.com/pricing>

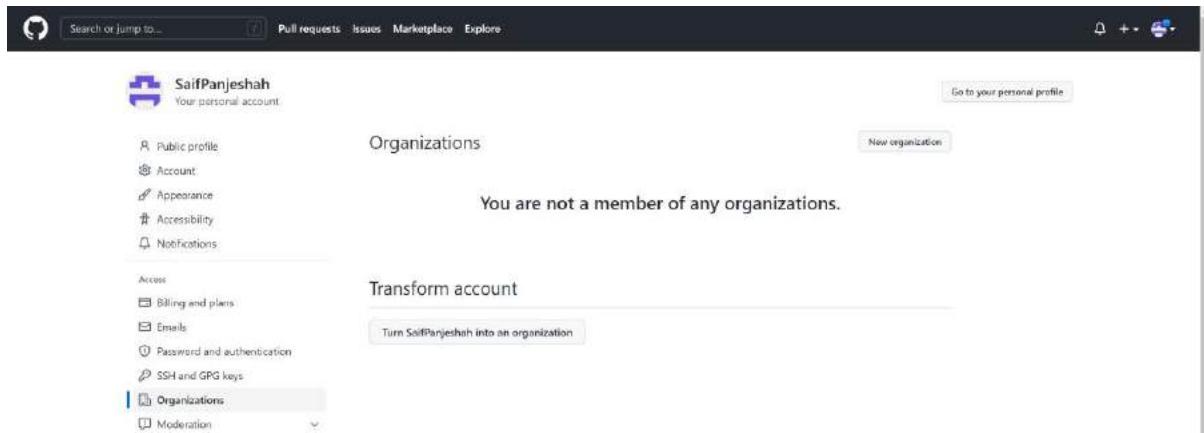


Fig. Personal GitHub Account

## Changing Repository Type from Public to Private:

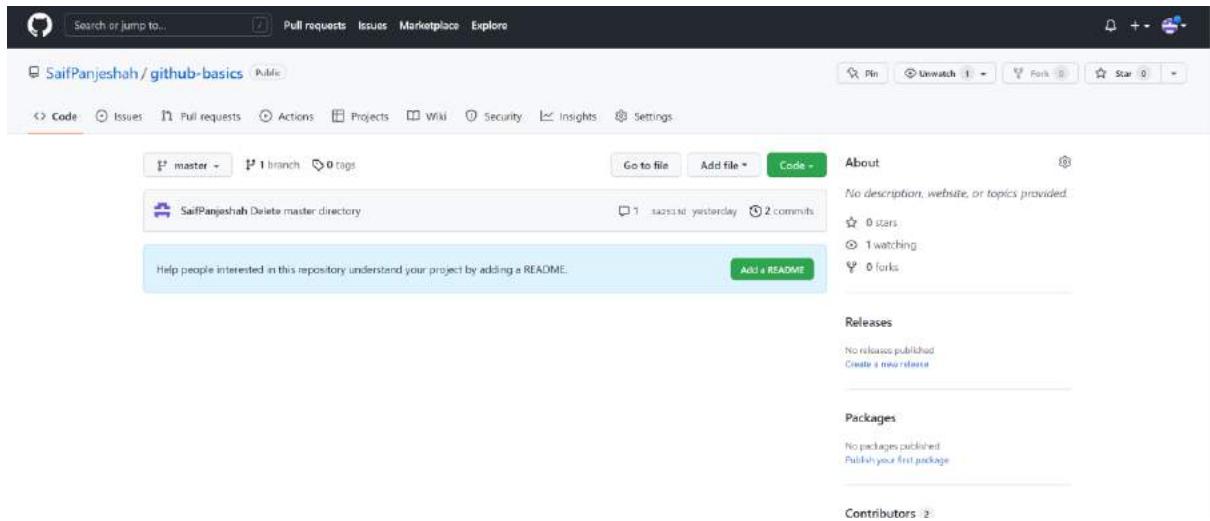


Fig. Shows Repository added in our account

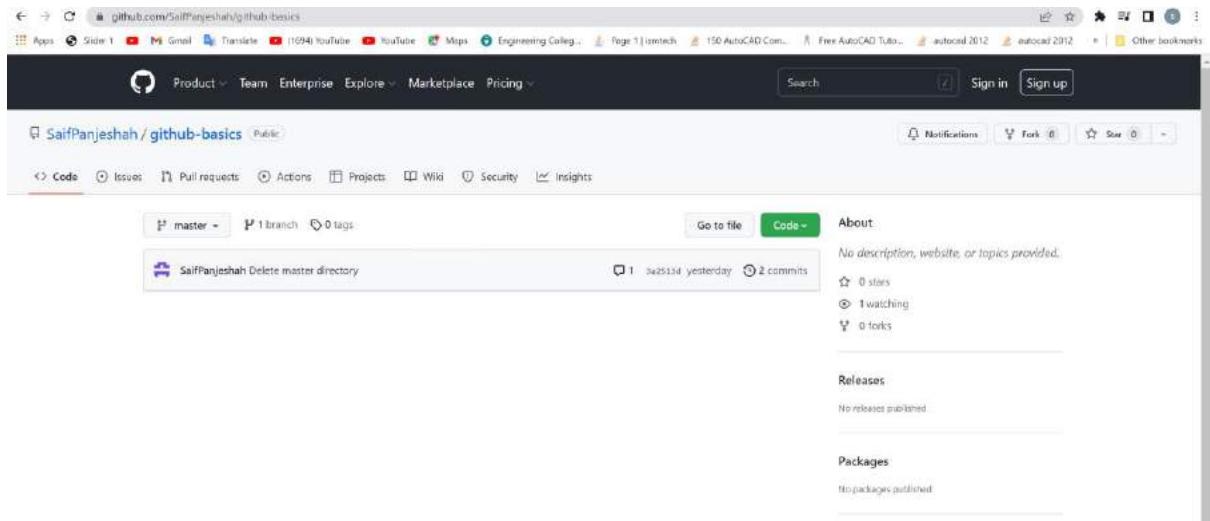
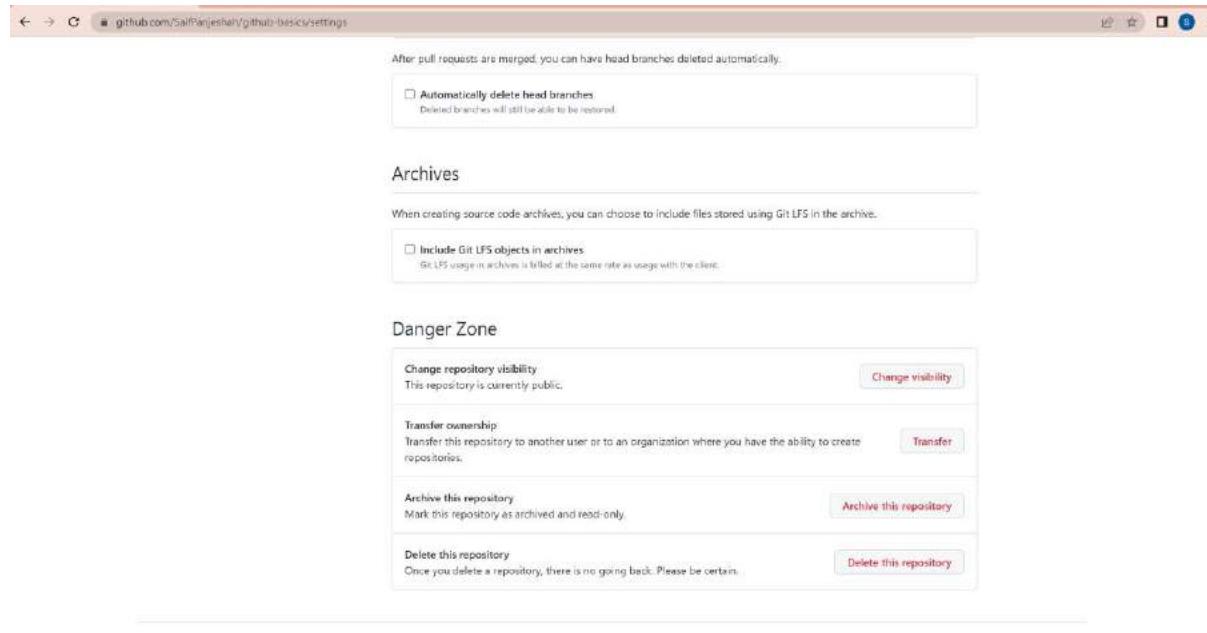


Fig. Accessing repositories from outside the browser or any user

**How Can we avoid so other the users can't get the access?**

**Go to the Setting and Change the repository visibility in Danger Zone to private**



**Fig. Visibility Setting of repositories**

Automatically delete head branches

Delete

Change repository visibility x

**Warning:** this is a potentially destructive action.

Make public  
This repository is currently public.

Make private  
Hide this repository from the public.

- You will **permanently** lose:
  - All 0 stars and 1 watcher of this repository.
  - All pages published from this repository.
- Dependency graph will remain enabled. Leaving them enabled grants us permission to perform read-only analysis on this repository.
- You can [upgrade your plan](#) to also avoid losing access to:
  - Codeowners functionality.
  - Any existing wikis.
  - Pulse, Contributors, Community, Traffic, Commits, Code Frequency and Network on the Insights page.
  - Draft PRs

Please type SaifPanjeshah/github-basics to confirm.

SaifPanjeshah/github-basics

[I understand, change repository visibility.](#)

[Change visibility](#)

[Archive this repository](#)

[Delete this repository](#)

[Terms](#) [Privacy](#) [Security](#) [Status](#) [Docs](#) [Contact GitHub](#) [Pricing](#) [API](#) [Training](#) [Blog](#) [About](#)



## Confirm access

Password

.....

[Forgot password?](#)

[Confirm password](#)

Tip: You are entering [sudo mode](#). We won't ask for your password again for a few hours.

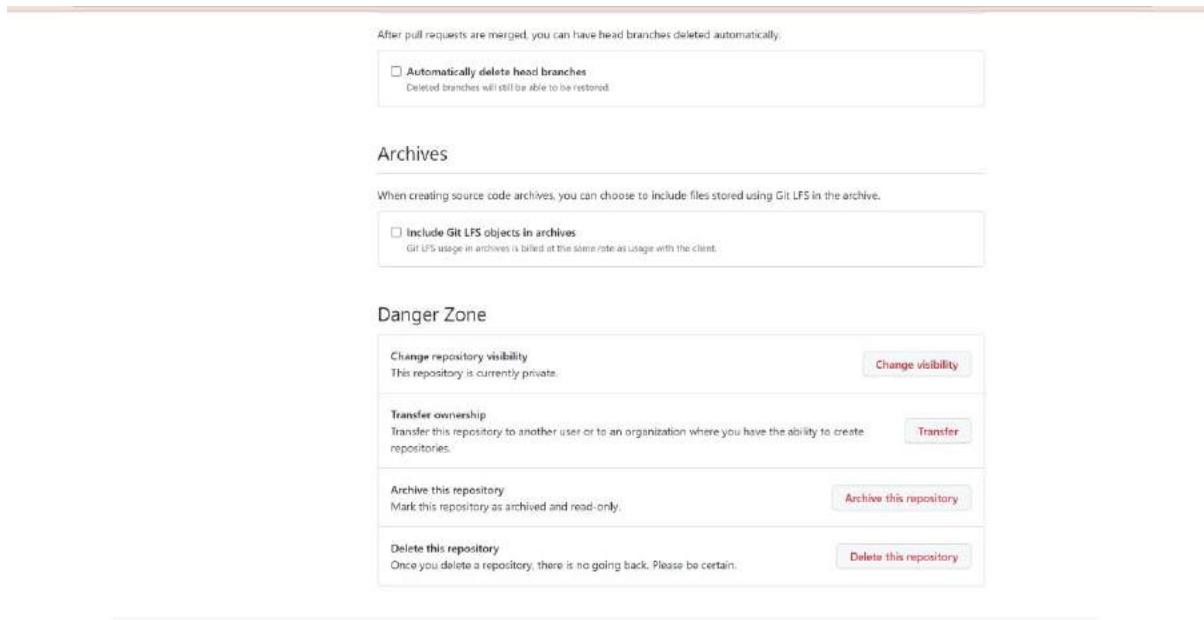


Fig. Successful Changes from public to private repository

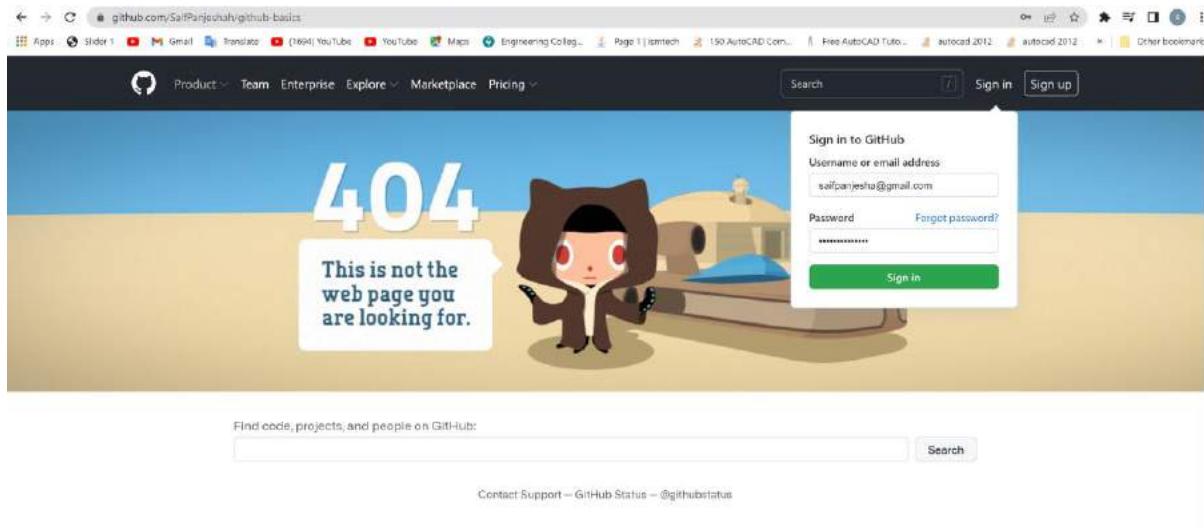


Fig. No one can access the repository

---

**Remember: For Changing the repository public follow the Steps Vice Versa.**

## Pushing Commits to Public Repository

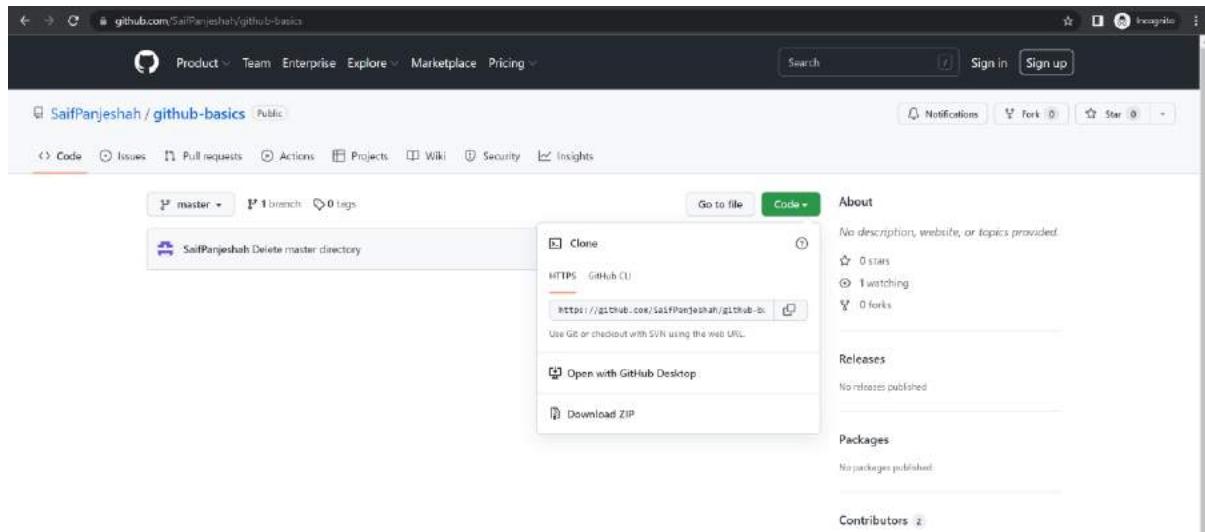
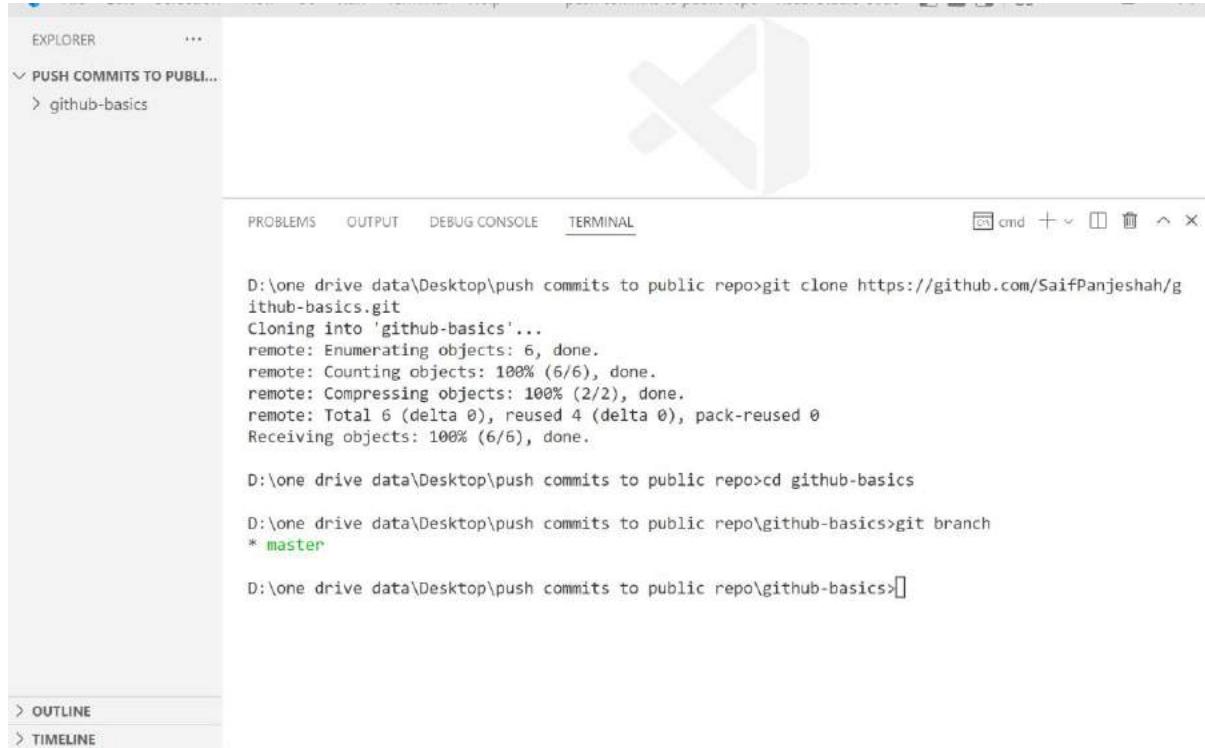


Fig. Changing the repository to public

**Copying the Http URL: <https://github.com/SaifPanjeshah/github-basics.git> and create new folder in vs code push the information our actual GitHub Project**



The screenshot shows the VS Code interface. In the Explorer sidebar, there is a folder named "github-basics". The terminal tab is active, displaying the following command-line session:

```
D:\One Drive\DATA\Desktop\push commits to public repo>git clone https://github.com/SaifPanjeshah/github-basics.git
Cloning into 'github-basics'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 6 (delta 0), reused 4 (delta 0), pack-reused 0
Receiving objects: 100% (6/6), done.

D:\One Drive\DATA\Desktop\push commits to public repo>cd github-basics

D:\One Drive\DATA\Desktop\push commits to public repo\github-basics>git branch
* master

D:\One Drive\DATA\Desktop\push commits to public repo\github-basics>
```

**Fig. Created new push commits to public folder**

Windows Credentials	Add a Windows credential
KR3	Modified: 11/04/2022
Certificate-Based Credentials	Add a certificate-based credential
No certificates.	
Generic Credentials	Add a generic credential
MSIX-Skype for Desktop MSAv2\live:saiffaizalpanjesh...	Modified: Today
MSIX-Skype for Desktop\live:saiffaizalpanjesh@101	Modified: Today
vscodevscode.github-authentication\github.auth	Modified: 22/05/2022
MicrosoftAccount\user=saiffaizalpanjesh@101@gmail.c...	Modified: Today
MicrosoftAccount\user=saiffaizalpanjesh@outlook.c...	Modified: Today
OneDrive Cached Credential	Modified: Today
virtualapp\didlogical	Modified: 30/04/2022

**Fig. deleting all GitHub credentials**

Github Apps  
OAuth Apps  
Personal access tokens

Personal access tokens

Generate new token  
Revoke all

Tokens you have generated that can be used to access the GitHub API.

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

Fig. deleted personal access tokens

The screenshot shows a terminal window with the following command history:

```
D:\One Drive\DATA\Desktop\push commits to public repo\github-basics>git add .  
D:\One Drive\DATA\Desktop\push commits to public repo\github-basics>git commit -m "pushes files on to commits to public repo"  
On branch master  
Your branch is up to date with 'origin/master'.  
nothing to commit, working tree clean  
D:\One Drive\DATA\Desktop\push commits to public repo\github-basics>git push  
fatal: credential-cache unavailable; no unix socket support  
fatal: credential-cache unavailable; no unix socket support  
Everything up-to-date  
D:\One Drive\DATA\Desktop\push commits to public repo\github-basics>git push origin master  
fatal: credential-cache unavailable; no unix socket support  
fatal: credential-cache unavailable; no unix socket support  
Everything up-to-date  
D:\One Drive\DATA\Desktop\push commits to public repo\github-basics>
```

Fig. Shows with access other users can't push the information our actual GitHub

## How GitHub Account Manager Security:



Fig. Added Account Manager Security

## Understanding & Adding Collaborator to a Private User Accounts:

The screenshot shows a VS Code interface with the following details:

- EXPLORER**: Shows a folder named "PUSH COMMITS TO PUBLIC..." containing "github-basics" and "push-to-public" branches. Under "push-to-public", there are files "push1.txt" and "pushfile.txt".
- TERMINAL**: Displays the command-line output of a git push attempt:

```
D:\One Drive Data\Desktop\push commits to public repo\github-basics>git add .  
D:\One Drive Data\Desktop\push commits to public repo\github-basics>git commit -m "pushes files on  
to commits to public repo"  
On branch master  
Your branch is up to date with 'origin/master'.  
nothing to commit, working tree clean  
D:\One Drive Data\Desktop\push commits to public repo\github-basics>git push  
fatal: credential-cache unavailable; no unix socket support  
fatal: credential-cache unavailable; no unix socket support  
Everything up-to-date  
D:\One Drive Data\Desktop\push commits to public repo\github-basics>git push origin master  
fatal: credential-cache unavailable; no unix socket support  
fatal: credential-cache unavailable; no unix socket support  
Everything up-to-date  
D:\One Drive Data\Desktop\push commits to public repo\github-basics>
```

Fig. Shows with access other users can't push the information our actual GitHub

To resolve this problem:

The screenshot shows the 'Who has access' section of a GitHub repository settings page. On the left, there's a sidebar with sections like 'Access', 'Code and automation', 'Security', and 'Integrations'. The 'Access' section is expanded, showing 'Collaborators' (selected), 'Moderation options', and other collapsed sections. The main area is titled 'Who has access' and contains two boxes: 'PUBLIC REPOSITORY' (which says 'This repository is public and visible to anyone.') and 'DIRECT ACCESS' (which says '0 collaborators have access to this repository. Only you can contribute to this repository.'). Below this is a 'Manage access' section with a message 'You haven't invited any collaborators yet' and a 'Add people' button.

Fig. Adding Collaborator to a Private User Accounts

This screenshot shows the same 'Who has access' section as the previous one, but now it displays a single pending invitation. The 'DIRECT ACCESS' box now says '1 has access to this repository. 0 collaborators. 1 invitation.' Below the 'Manage access' section, there's a table listing the pending invitee: 'SaifPanjesha' (with a pending invite icon) and 'Awaiting SaifPanjesha's response'. There are checkboxes for selecting all and removing the invite, and a search bar for finding a collaborator.

Fig. Pending User Invitation

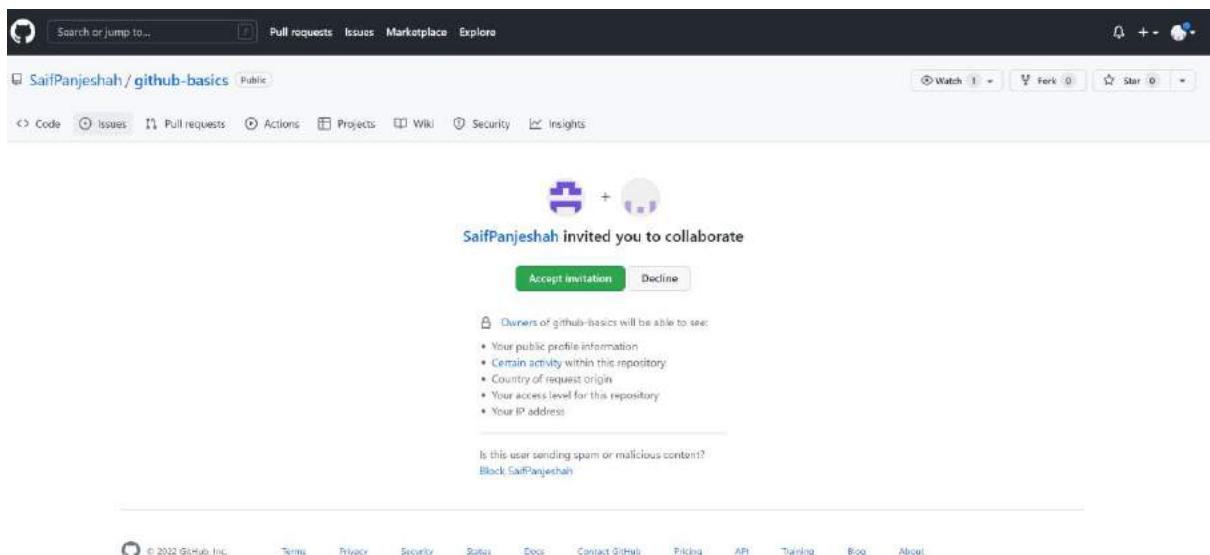


Fig. Accepting User request from another user account

### Who has access

This screenshot shows the "Who has access" section for a repository. It is divided into two main sections: "PUBLIC REPOSITORY" and "DIRECT ACCESS".  
The "PUBLIC REPOSITORY" section contains the text: "This repository is public and visible to anyone." and a "Manage" button.  
The "DIRECT ACCESS" section contains the text: "1 has access to this repository. 1 collaborator." and a "Remove" button.

### Manage access

This screenshot shows the "Manage access" interface. It includes a search bar labeled "Find a collaborator...", a list of users with checkboxes, and a "Remove" button next to each user entry. The user listed is "SaifPanjeshah" with the role "Collaborator". There are also "Select all" and "Type" dropdown buttons at the top of the list.

Fig. Successful added

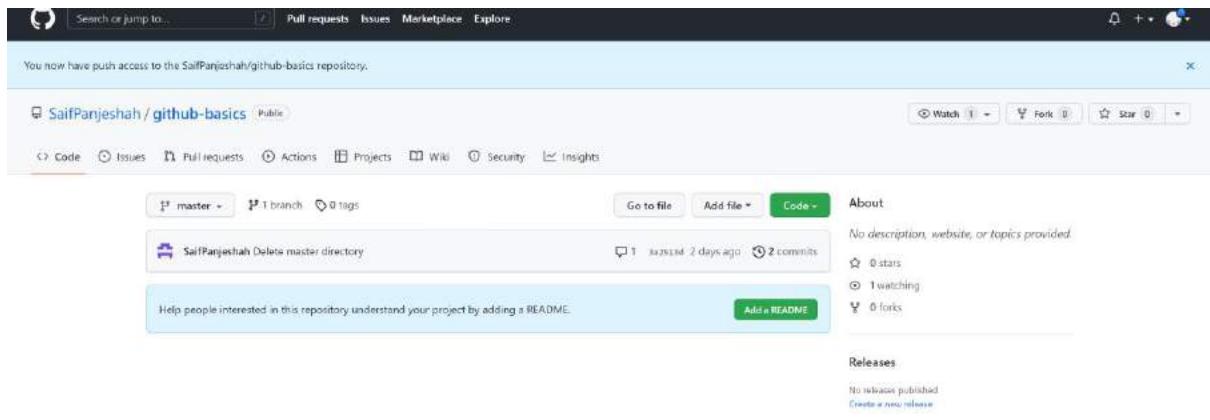


Fig. push access to another users

Now, here the problem is even with access users can't push the information our actual GitHub. because of personal access token and we as account owner can't share the personal access token.

**Remember:** It's better to get personal access token requests from another users (Collaborators)

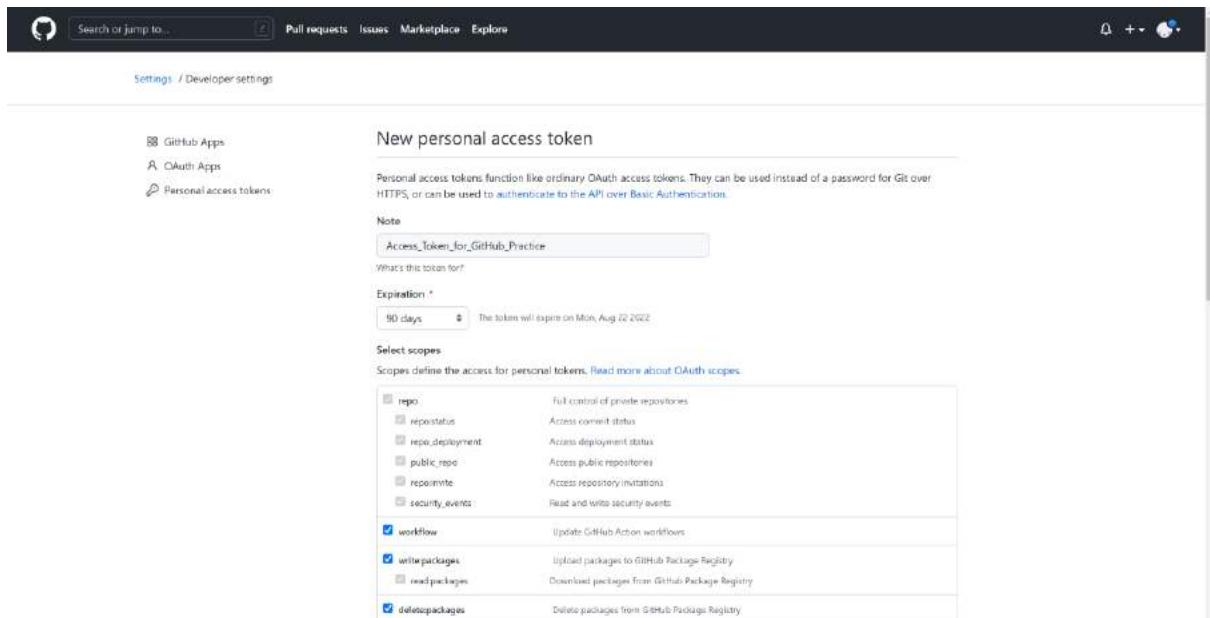


Fig. creating access Token From another user accounts

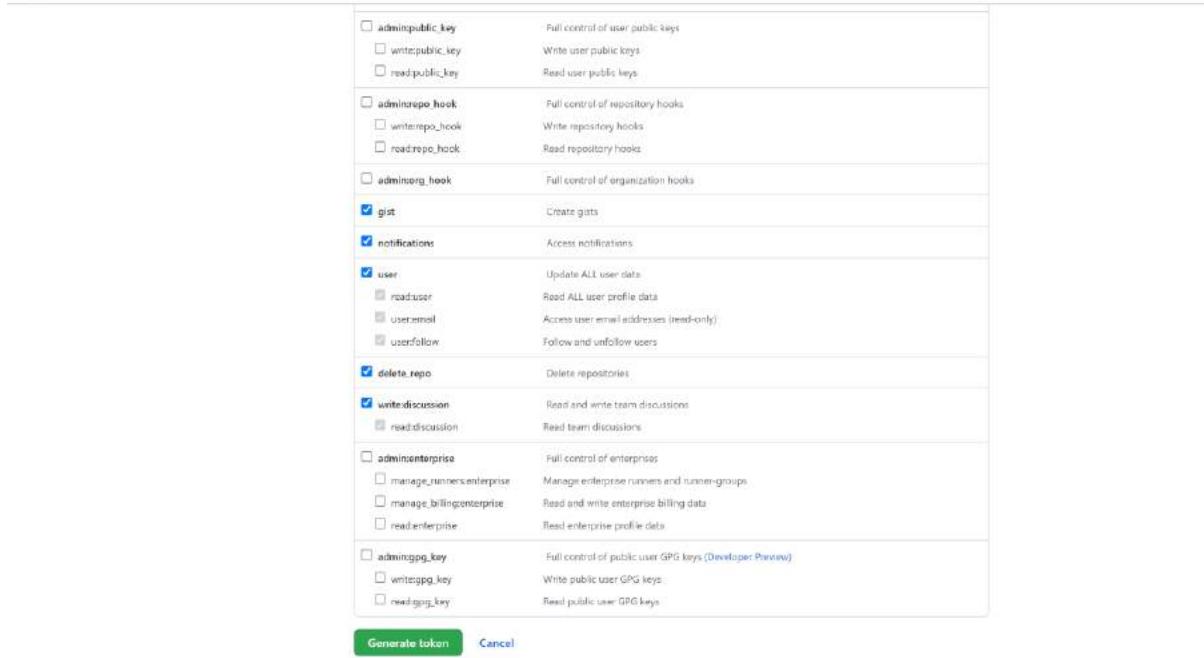


Fig. Creation Successful generate the token now without admin access

## Created new file1.txt from local tracking branch of another user

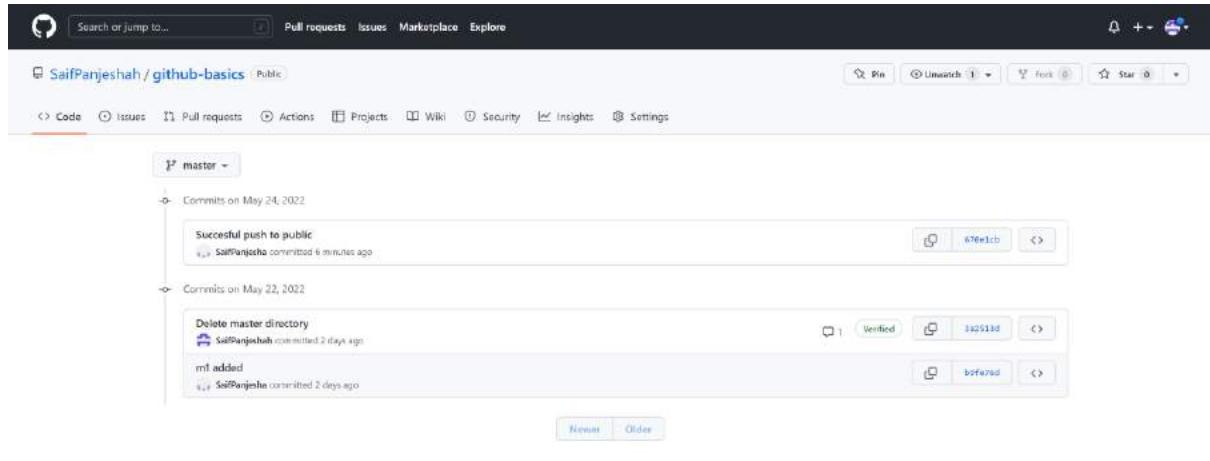
```

D:\One Drive\DATA\Desktop\push commits to public repo\github-basics>git add .
D:\One Drive\DATA\Desktop\push commits to public repo\github-basics>git commit -m "Successful push to public"
[master 670e1cb] Successful push to public
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file1.txt

D:\One Drive\DATA\Desktop\push commits to public repo\github-basics>git push
fatal: credential-cache unavailable; no unix socket support
fatal: credential-cache unavailable; no unix socket support
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Writing objects: 100% (3/3), 273 bytes | 273.00 KiB/s, done.
Total 3 (delta 0), reused 1 (delta 0), pack-reused 0
To https://github.com/SaifPanjeshah/github-basics.git
 3a2513d..670e1cb master -> master
  
```

Fig. Succesful push the changes in account

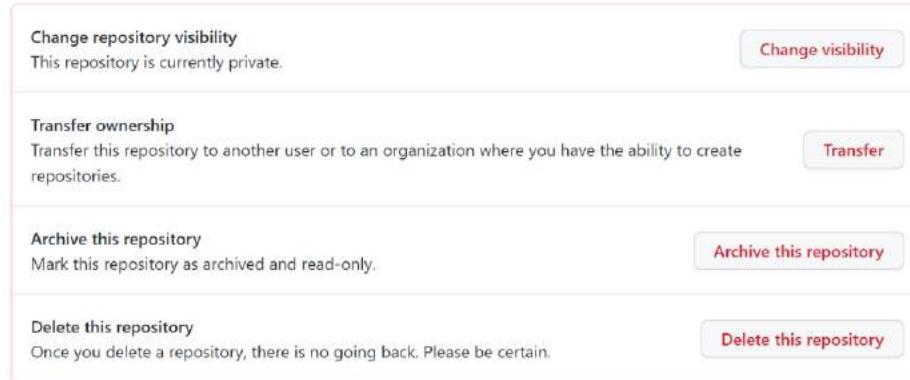
**To view please check the owner GitHub account**



**Fig. Commit Succesful added**

## **Collaborating in Private repositories:**

### **Danger Zone**



**Fig. Changing repositories in private**

The screenshot shows the VS Code interface with the following details:

- EXPLORER**: Shows a tree view of files and folders. A file named "push1.txt" is selected under the "push-to-public" folder.
- TERMINAL**: Displays the command-line output of the git push process. The output shows the commit message "Private push repo!", the number of changes (1 file changed), and the creation mode of the file.
- OUTPUT**: Shows the command "git add .".
- PROBLEMS**: Shows the command "git commit -m "Private push repo!"".
- DEBUG CONSOLE**: Not visible in the screenshot.
- COMMAND PALETTES**: Shows the command "cmd" followed by a series of icons for copy, paste, find, and others.

```

D:\OneDrive\data\Desktop\push commits to public repo\github-basics>git add .
D:\OneDrive\data\Desktop\push commits to public repo\github-basics>git commit -m "Private push repo!"
[master 9ee294c] Private push repo!
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 push-to-public/push1.txt

D:\OneDrive\data\Desktop\push commits to public repo\github-basics>git push
fatal: credential-cache unavailable; no unix socket support
fatal: credential-cache unavailable; no unix socket support
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 338 bytes | 338.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/SaiPanjeshah/github-basics.git
    7298781..9ee294c  master -> master

D:\OneDrive\data\Desktop\push commits to public repo\github-basics>

```

Fig. Successful push to private repository

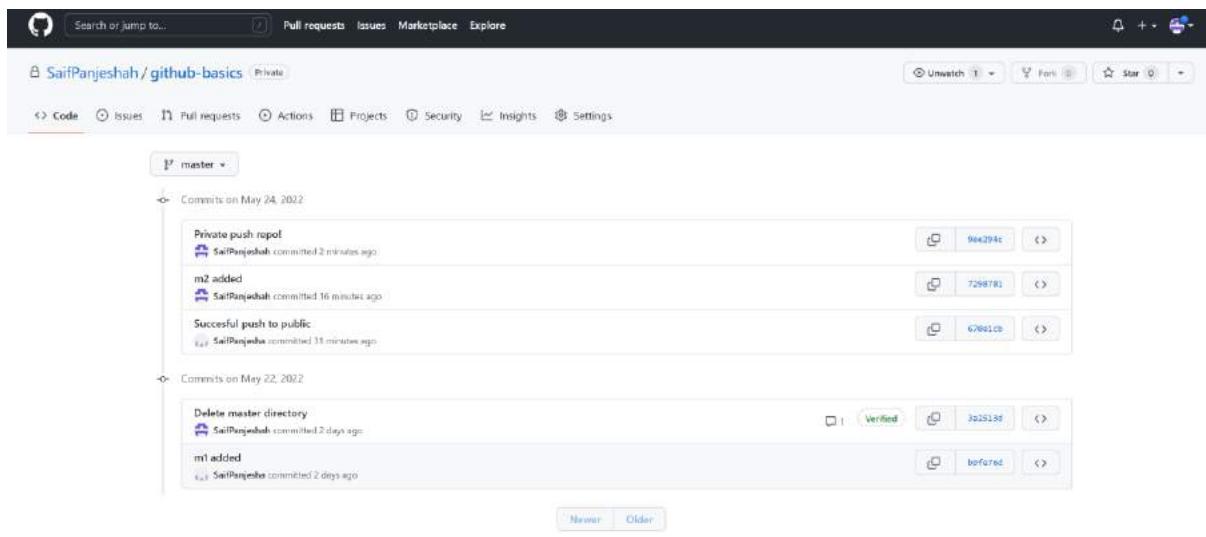


Fig. Successful push to private repository in GitHub

## Comparing Owner & Collaborator Rights:

### Official Site:

<https://docs.github.com/en/account-and-profile/setting-up-and-managing-your-personal-account-on-github/managing-personal-account-settings/permission-levels-for-a-personal-account-repository>

### Limit Interactions:

Providing restrictions to another user.

Path: Goto Profile -> Settings -> Moderation

The screenshot shows the GitHub user profile page for 'SaifPanjeshah'. The navigation bar at the top includes links for 'Search or jump to...', 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. On the right, there's a link to 'Go to your personal profile'. The main sidebar on the left has sections like 'Public profile', 'Account', 'Appearance', 'Accessibility', 'Notifications', 'Access', 'Billing and plans', 'Emails', 'Password and authentication', 'SSH and GPG keys', 'Organizations', and 'Moderation'. Under 'Moderation', the 'Blocked users' option is selected. The main content area is titled 'Block a user' and contains instructions about what happens when a user is blocked. It also lists what blocked users are not able to do. A search bar is provided to find specific users. Below this is a section titled 'Blocked users' which displays the message 'You have not blocked any users.' There is also a checkbox for 'Warn me when a blocked user is a prior contributor to a repository' with a note explaining it only applies to repositories the user hasn't contributed to yet.

Fig. Block a user's

The screenshot shows the GitHub account settings page for a user named SaifPanjeshah. The left sidebar has a 'Temporary interaction limits' section selected under 'Interaction limits'. This section contains three configuration boxes: 'Limit to existing users', 'Limit to prior contributors', and 'Limit to repository collaborators'. Each box includes checkboxes for 'New users' (unchecked), 'Users' (checked), 'Contributors' (checked), and 'Collaborators' (checked). Buttons for 'Enable' and 'Disable' are shown next to each box.

Fig. Limit Iteration

The screenshot shows the settings page for a repository named 'github-basics'. The 'General' tab is selected. Under 'Repository name', the value 'github-basics' is displayed with a 'Rename' button. A checkbox for 'Template repository' is present with a descriptive note. The 'Social preview' section allows for uploading an image, with a note about file size and resolution, and a 'Download template' link. The left sidebar lists repository management options like 'Code', 'Issues', 'Pull requests', 'Actions', 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings'.

Fig. Changes repositories to public for applying Limit Iteration

**Note : We cannot apply Limit to specific user only only on general conditions that apply to certain users.**

**Code Review Limit:** Restrict users who are permitted to approve or request changes on pull requests in this repository.

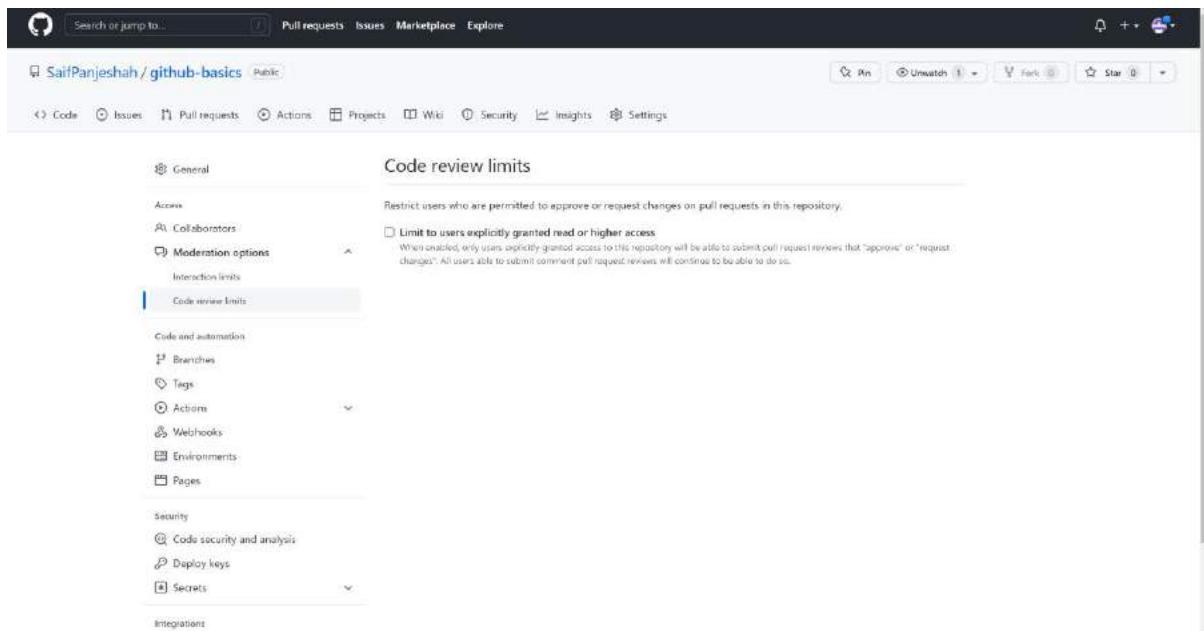


Fig. Code review limits

## Introducing Organizations:

we have some limits here when it comes to managing specific access rights for collaborators of our projects



Fig. Added Account Manager Security

The big picture here of the whole security part. So, if you work in smaller projects with personal user accounts and some collaborators, you can perfectly do that. If you are part of a bigger project, or if you want to manage the specific rights the members of your project have, then an organization account might be the way to go.

### Creating an Organization Account:

Go to Profile -> Setting -> Organizations -> Click on New Organization

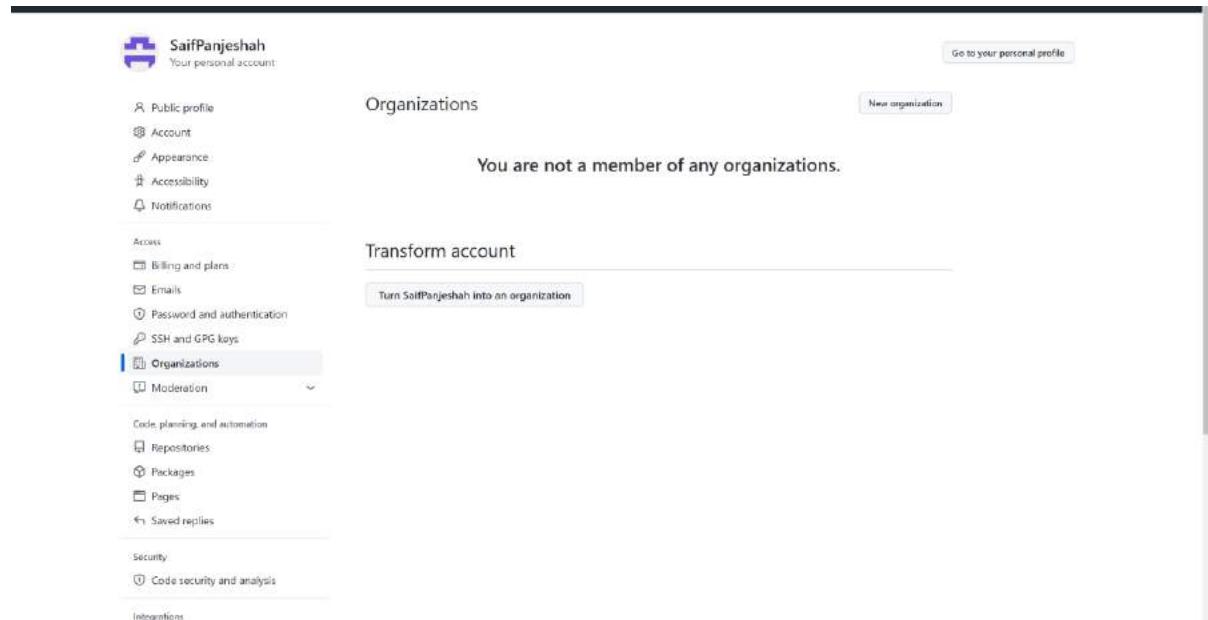


Fig. Creating organization account

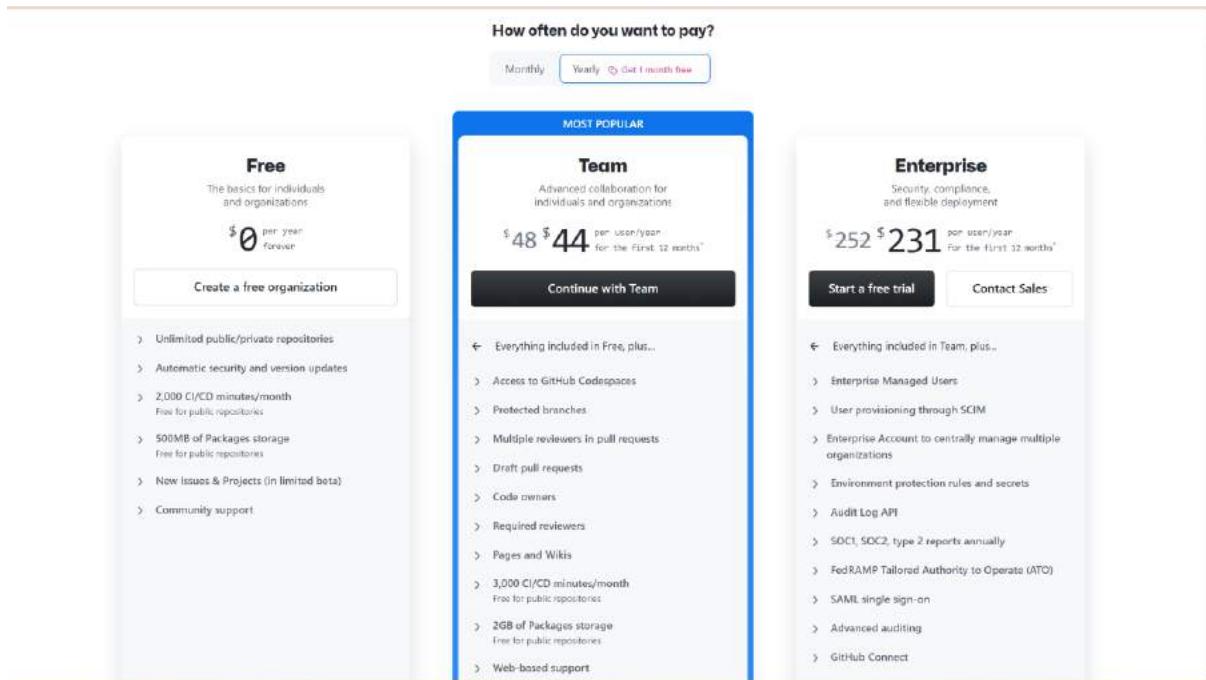


Fig. Pricing Start as Free Organization

This screenshot shows the "Create a free organization" form. It includes fields for "Organization account name" (set to "Commitsa"), "Contact email", and "This organization belongs to:" (radio buttons for "My personal account" (selected) and "A business or institution"). Below these is a "Verify your account" section containing a CAPTCHA puzzle and a "Verify" button. At the bottom, there is a checkbox for accepting the Terms of Service and a large green "Next" button.

Fig. Adding Organization Name Commitsa



Fig. Verified Successful

A screenshot of the GitHub organization setup page. The title 'Welcome to GitHub' is at the top. Below it, a message says 'Woohoo! You've joined millions of developers who are doing their best work on GitHub. Tell us what you're interested in. We'll help you get there.' There are several sections for input:

- 'Tell us about you': A section asking 'What do you spend time on most day-to-day?' with a note 'Please select all that apply.' It includes four options with checkboxes: 'Writing code', 'Managing and coordinating engineering work', 'Planning projects', and 'Billing administration'.
- 'Other': A text input field labeled 'Please describe'.
- 'Tell us about your team': A section asking 'How many people do you expect to actively work within this GitHub organization?' with a note 'Please select all that apply.' It includes five radio buttons: '0', '1-5', '6-15', '16-24', and '25+'.
- 'What type of work do you plan to use this organization for?': A section with a note 'Please select all that apply.' It includes a list of items with checkboxes, though they are not fully visible in the screenshot.

Fig. Complete the Setup and submit

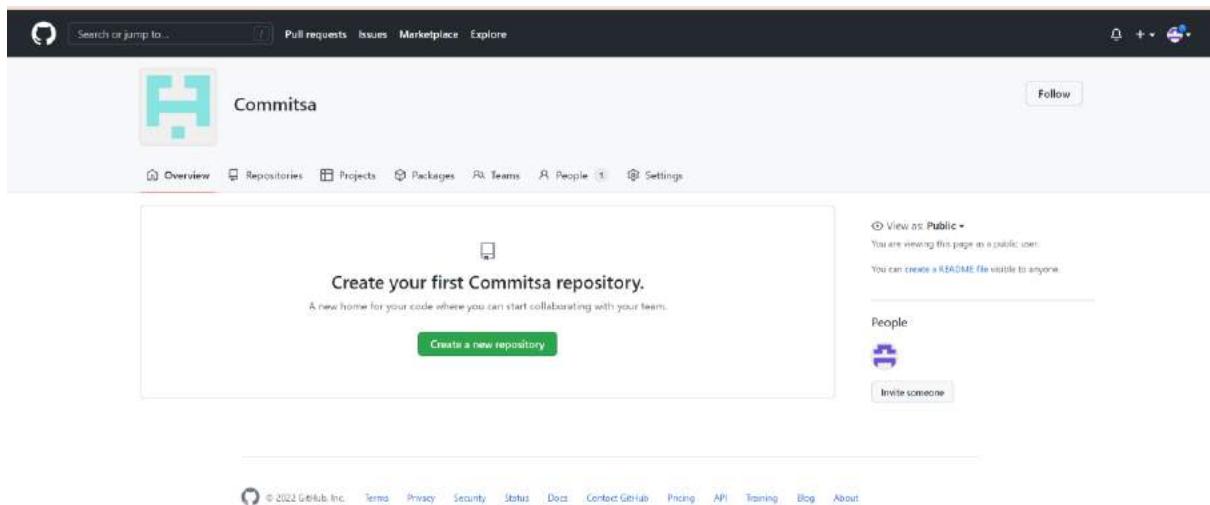


Fig. Successful Created

## To View Organization from Dashboard

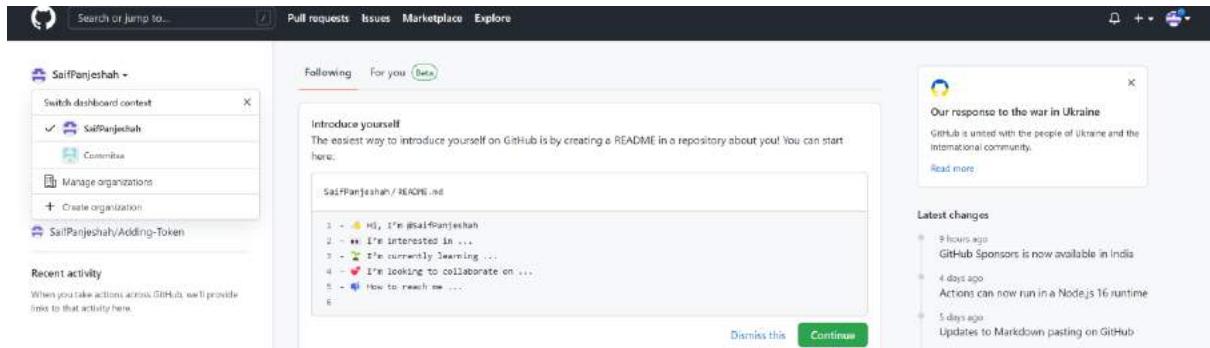


Fig. View Organization

## Exploring Member Repository Permission:

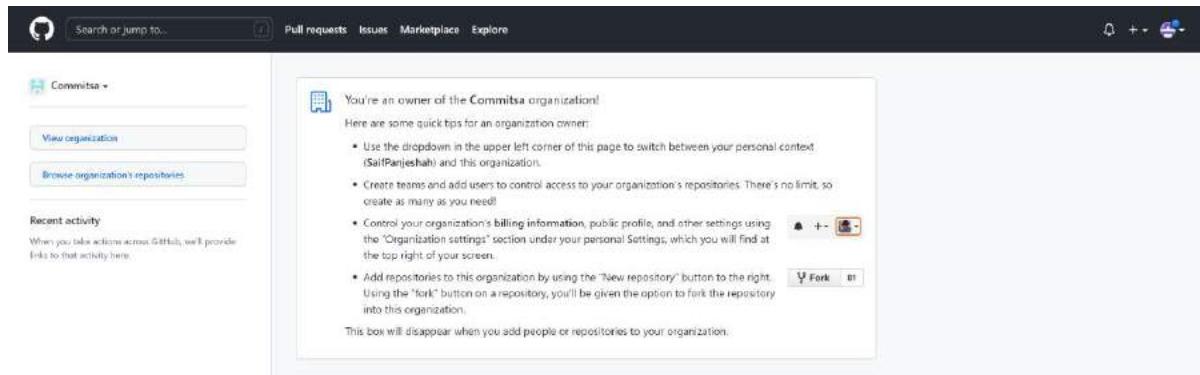


Fig. Commitsa Organization

## What Missing, Here Member and repositories in Organization?

### Creating New Repository:

Path: Inside Organization -> Create New Repo

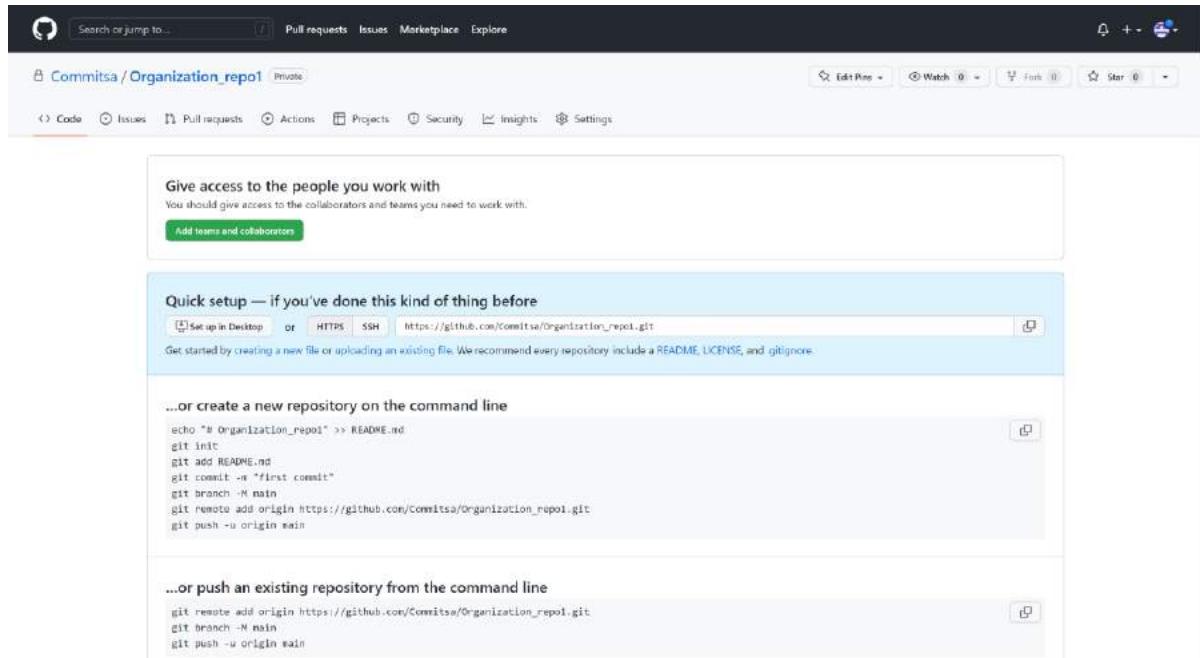


Fig. Successful Created Private Repository in Organization

## After Creating repository Inviting Teams Members Same as Personal Account:

The screenshot shows the 'Who has access' section of a GitHub repository settings page. On the left, a sidebar lists various access categories like General, Collaborators and teams, and Security. Under 'Collaborators and teams', it says 'PRIVATE REPOSITORY' and 'Only those with access to this repository can view it.' It shows one member with a 'Read' role. In the center, there's a 'Manage' button. To the right, under 'DIRECT ACCESS', it says '0 teams or members have access to this repository. Only Owners can contribute to this repository.' Below this, a 'Manage access' section shows a message: 'You haven't added any teams or people yet'. It includes 'Add people' and 'Add teams' buttons.

Fig. Manage access

We can privilege our base role member (account owner)

The screenshot shows the 'Member privileges' section of an organization account settings page. On the left, a sidebar lists categories like General, Access, Billing and plans, Member privileges (which is selected), and others. Under 'Member privileges', it shows 'Base permissions' with a note about inheritance. It has a 'Read' dropdown set to 'Read'. Below that are sections for 'Repository creation' (with 'Public' and 'Private' options), 'Repository forking' (with a note about enabling forked repos), and 'Pages creation' (with 'Public' and 'Private' options). Each section has a 'Save' button at the bottom.

Fig. Base role privileges member

## Adding Outside Collaborator:

An *outside collaborator* is a person who is not a member of your organization, but has access to one or more of your organization's repositories.

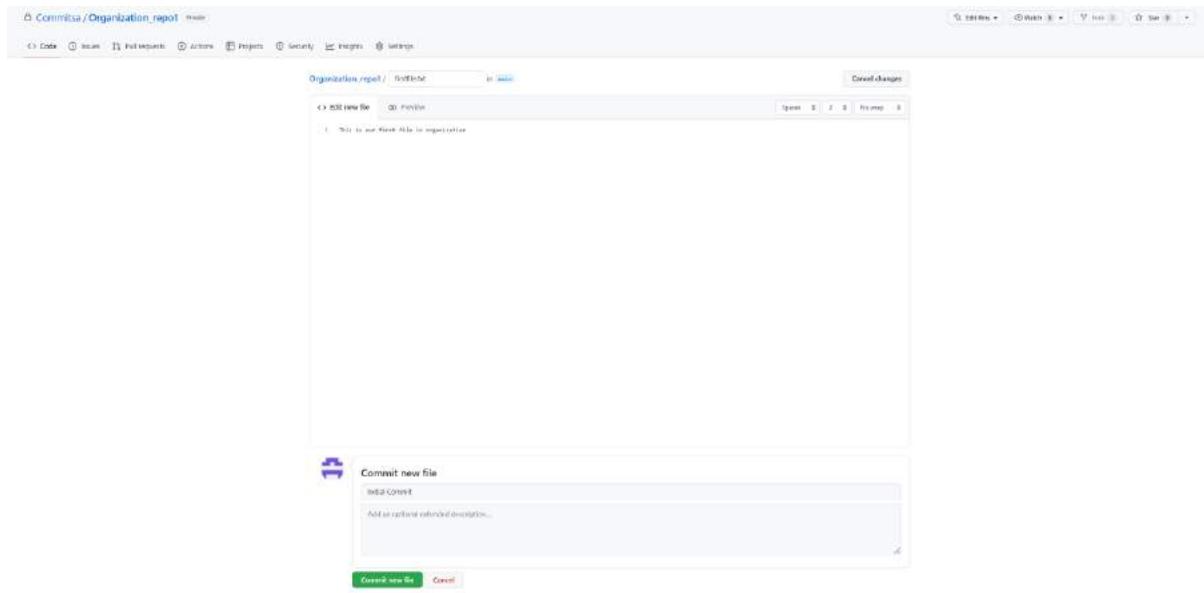


Fig. Adding First File in Organizational Account

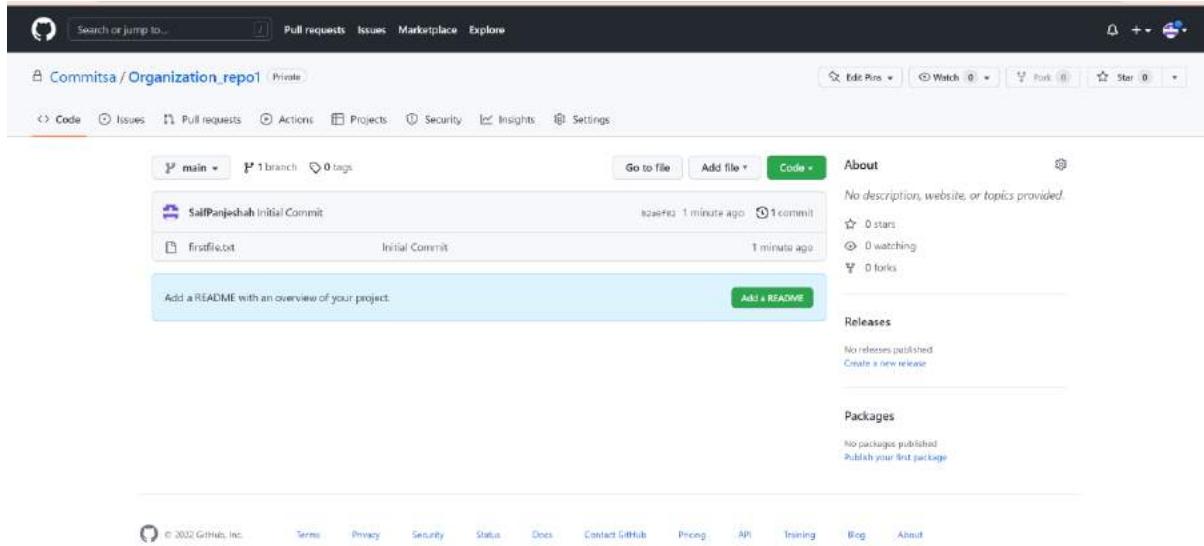


Fig. FirstFile.txt Created

The screenshot shows the 'Access' section of a GitHub repository settings page. The 'Collaborators and teams' tab is selected. On the left, there's a sidebar with options like 'Code and automation', 'Security', and 'Integrations'. The main area is titled 'Who has access' and shows three sections: 'PRIVATE REPOSITORY' (Read), 'BASE ROLE' (Read), and 'DIRECT ACCESS' (Read). It indicates that 1 member has access. Below this is a 'Manage' button. A large central box is titled 'Manage access' with a sub-section 'Add people' containing a search bar and a green 'Select a member above' button.

Fig. Collaborators and teams settings

This screenshot shows the 'Add people' modal window from the previous figure. The modal title is 'Add people to Organization\_repo1'. It contains a search bar with placeholder text 'Search by username, full name, or email' and a large green button labeled 'Select a member above'. The background of the main interface is dimmed, and the 'Manage access' section is visible at the bottom.

Fig. Adding Outside Collaborators Organization read only permission

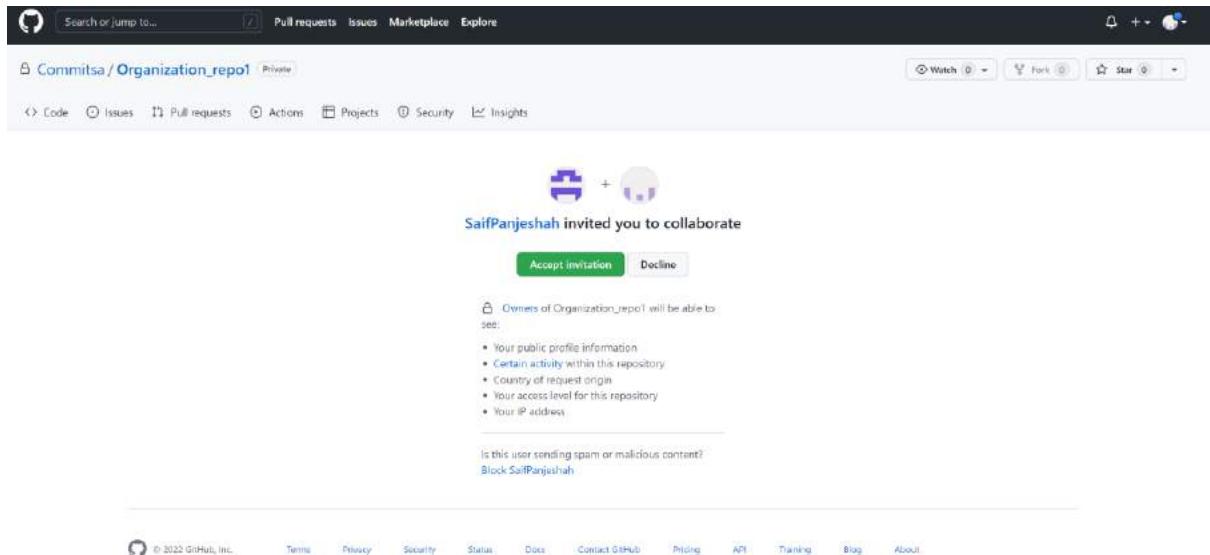


Fig. Outside Collaborators Organization Invitation

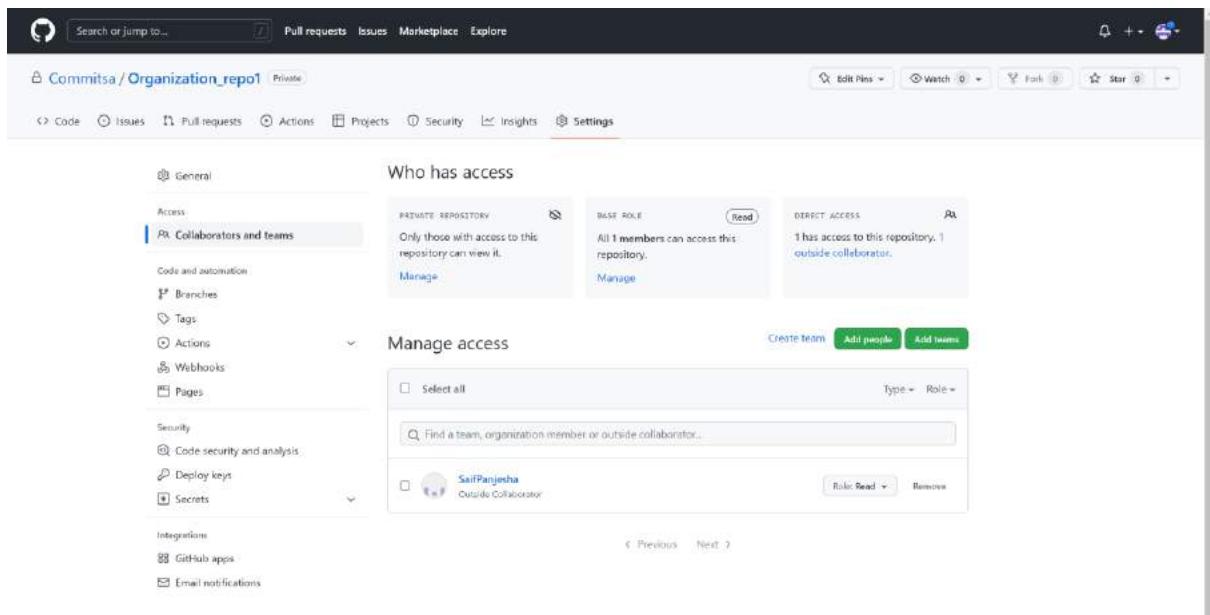


Fig. Successful added outside collaborator

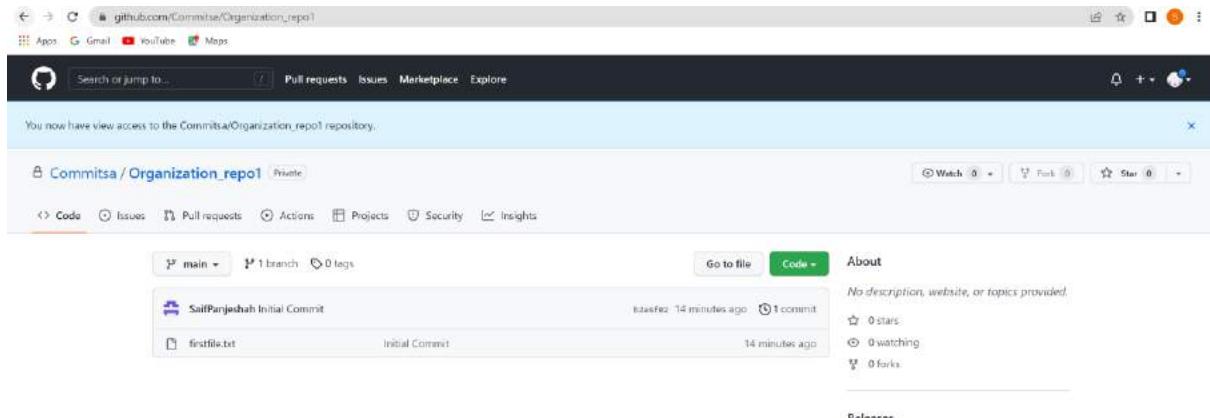


Fig. Given view access permission only

Now, trying to push the code from local branch git (VS Code) to check whether the code is successful to cloud or not.

**git clone [https://github.com/Commitsa/Organization\\_repo1.git](https://github.com/Commitsa/Organization_repo1.git) .**



Fig. Successful created organization files

```

D:\One Drive Data\Desktop\Organization>git clone https://github.com/Commitsa/Organization_repo1.git .
Cloning into '.'...
fatal: credential-cache unavailable; no unix socket support
fatal: credential-cache unavailable; no unix socket support
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.

D:\One Drive Data\Desktop\Organization>git add .

D:\One Drive Data\Desktop\Organization>git commit -m "Second File added from outside collaborators"
[main 7c93986] Second File added from outside collaborators
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 secondfile.txt

D:\One Drive Data\Desktop\Organization>git push
fatal: credential-cache unavailable; no unix socket support
fatal: credential-cache unavailable; no unix socket support
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 306 bytes | 306.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Commitsa/Organization_repo1.git
 b2a6f02..7c93986 main -> main

D:\One Drive Data\Desktop\Organization>

```

Fig. Successful added to organization file

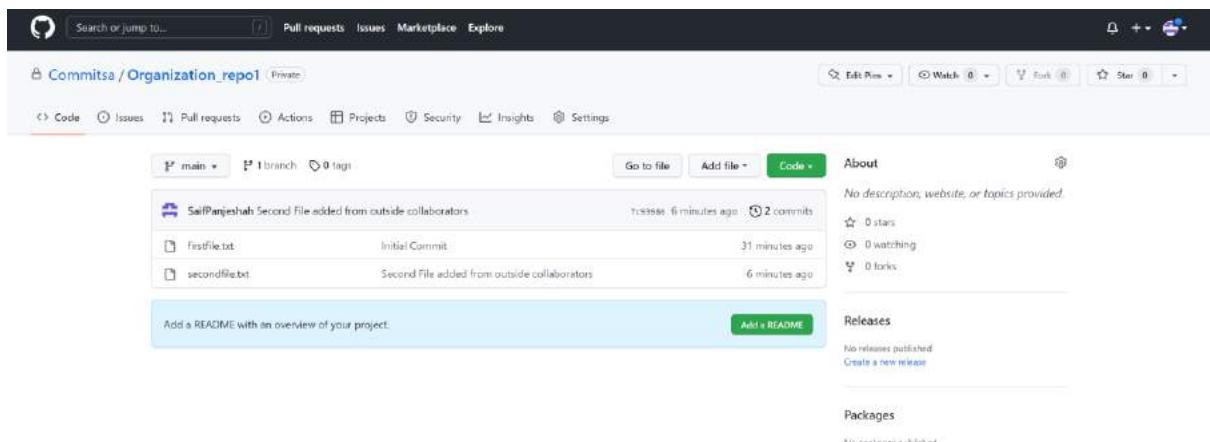


Fig. Successful added to GitHub

## Adding Organization Members:

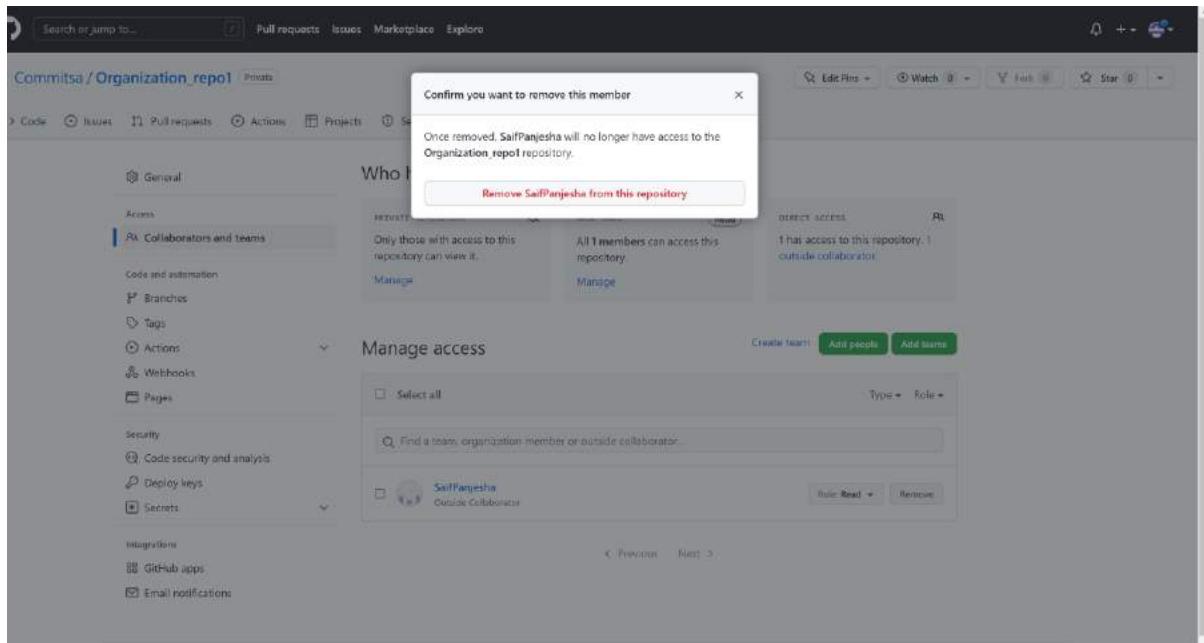


Fig. removing the outside collaborators

## Let's explore people tab on Organization:

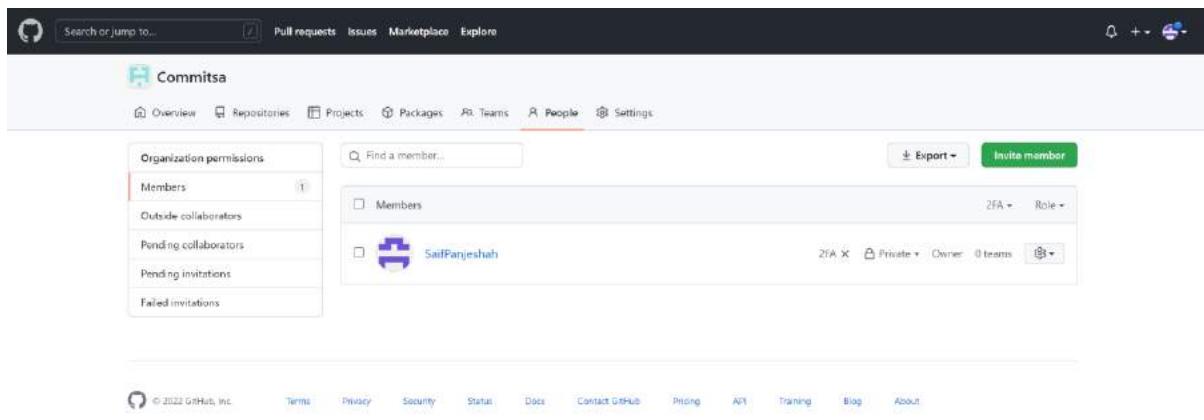


Fig. Explore people tab

## Inviting Member:

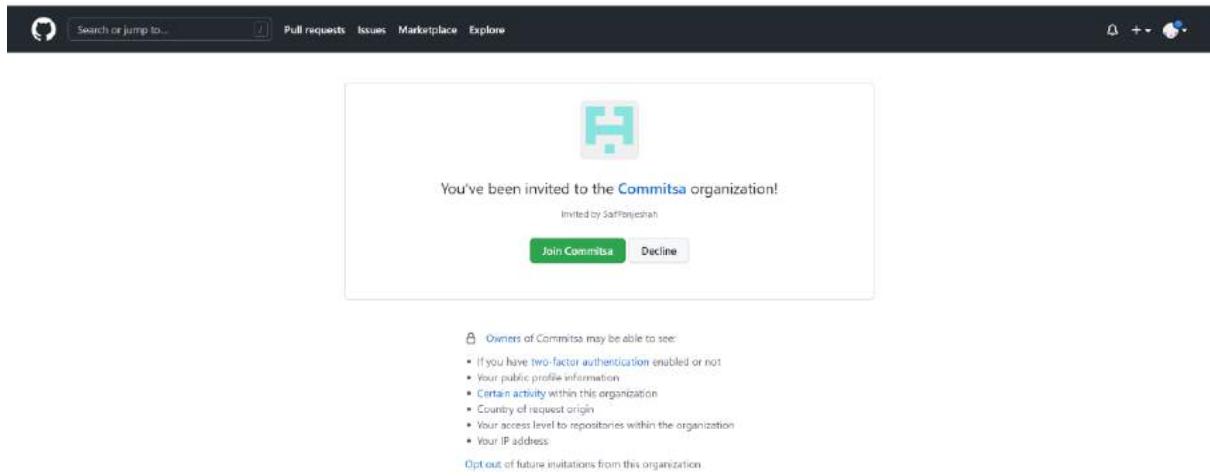


Fig. Inviting member to Commitsa organization

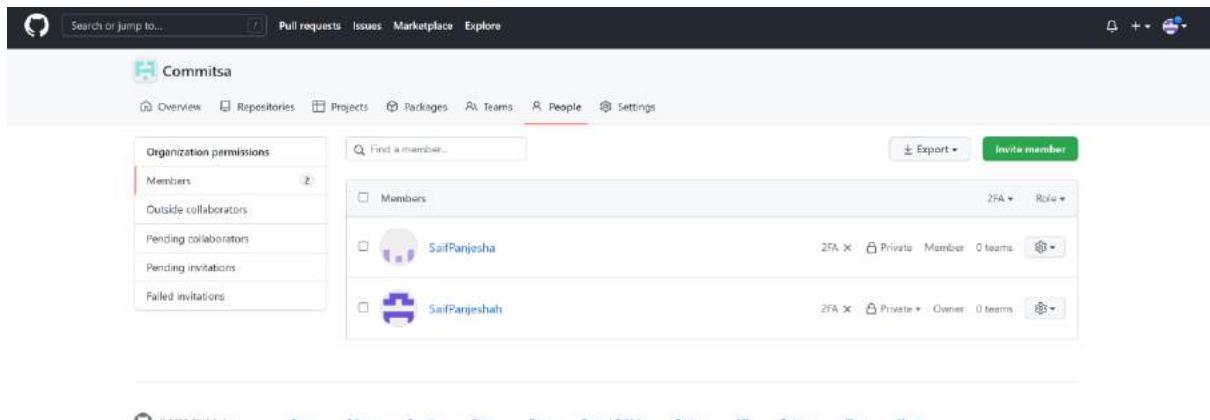
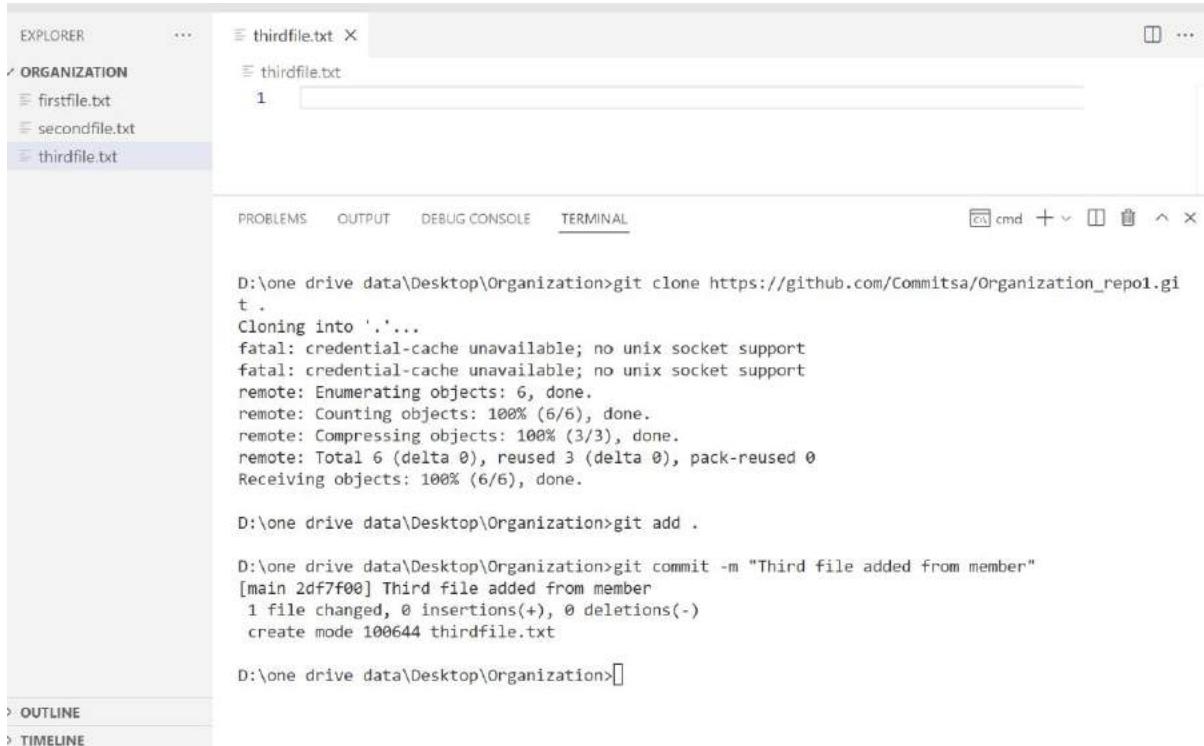


Fig. Successful Invitation

**Now, trying to push the code from local branch git (VS Code) to check whether the code is successful to cloud or not.**



The screenshot shows the VS Code interface with the terminal tab selected. The command history in the terminal window is as follows:

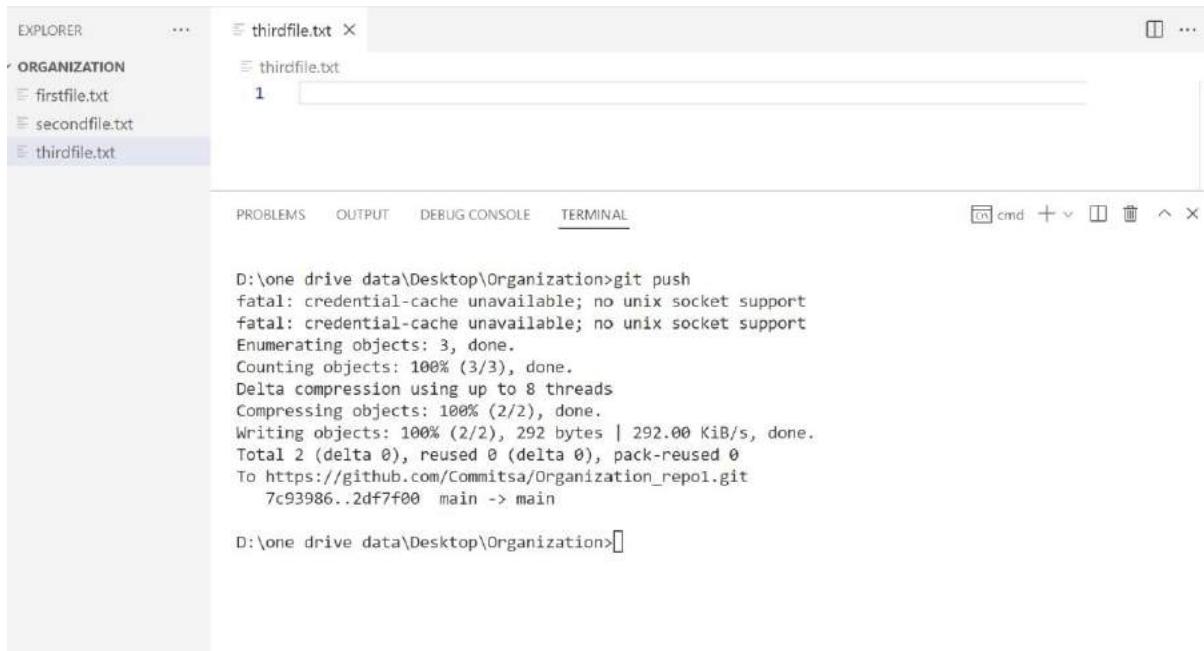
```
D:\one drive data\Desktop\Organization>git clone https://github.com/Commitsa/Organization_repo1.git.
Cloning into '.'...
fatal: credential-cache unavailable; no unix socket support
fatal: credential-cache unavailable; no unix socket support
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 6 (delta 0), reused 3 (delta 0), pack-reused 0
Receiving objects: 100% (6/6), done.

D:\one drive data\Desktop\Organization>git add .

D:\one drive data\Desktop\Organization>git commit -m "Third file added from member"
[main 2df7f00] Third file added from member
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 thirdfile.txt

D:\one drive data\Desktop\Organization>
```

**Fig. Successful clone**



The screenshot shows the VS Code interface with the terminal tab selected. The command history in the terminal window is as follows:

```
D:\one drive data\Desktop\Organization>git push
fatal: credential-cache unavailable; no unix socket support
fatal: credential-cache unavailable; no unix socket support
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 292 bytes | 292.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Commitsa/Organization_repo1.git
 7c93986..2df7f00 main -> main

D:\one drive data\Desktop\Organization>
```

**Fig. Successful Push**

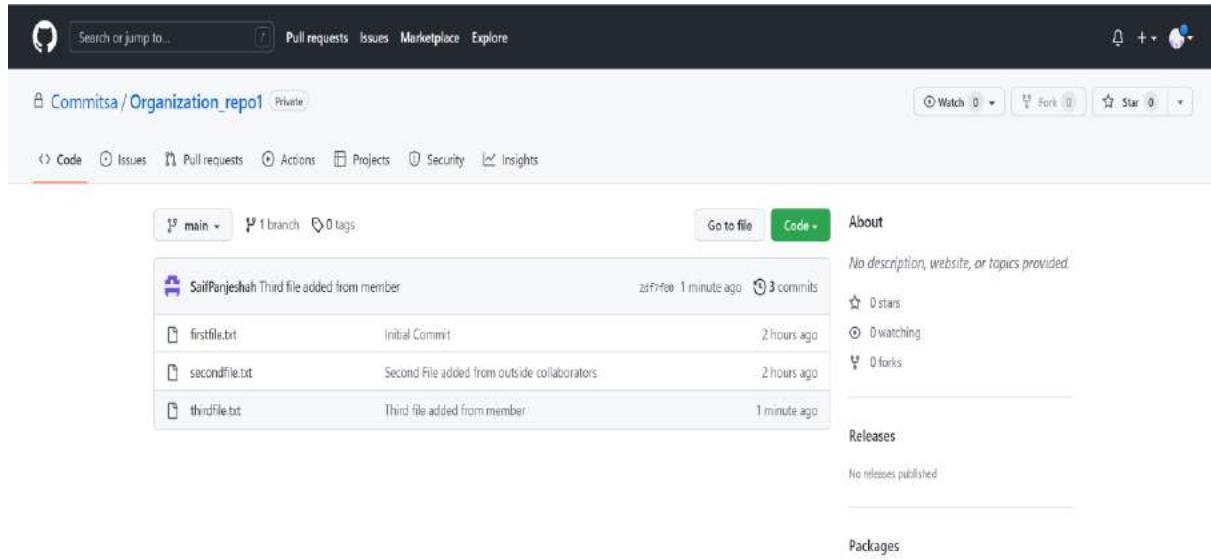


Fig. Successful Push on GitHub

## Failing to Manage Access for Individual Repositories:

The screenshot shows the 'Member privileges' section of the Commitsa organization account settings. The left sidebar has 'Member privileges' selected under 'Access'. The main content area is titled 'Member privileges' and contains sections for 'Base permissions', 'Repository creation', 'Repository forking', and 'Pages creation'. Under 'Base permissions', it says 'Base permissions to the organization's repositories apply to all members and excludes outside collaborators. Since organization members can have permissions from multiple sources, members and collaborators who have been granted a higher level of access than the base permissions will retain their higher permission privileges.' There are two options: 'Public' (unchecked) and 'Private' (checked). A 'Save' button is present. Under 'Repository creation', it says 'Members will be able to create only selected repository types. Outside collaborators can never create repositories.' There are two options: 'Public' (unchecked) and 'Private' (checked). A 'Save' button is present. Under 'Repository forking', it says 'Allow forking of private repositories' (unchecked). A 'Save' button is present. Under 'Pages creation', it says 'Members will be able to publish sites with only the selected access controls.' There are two options: 'Public' (checked) and 'Private' (unchecked). A 'Save' button is present.

Fig. Giving Member privileges access to Write Option

## Creating New Repositories in Organizational Account for read only members:

The screenshot shows a GitHub repository named 'member\_read\_only'. The user is logged in as 'member'. A commit dialog is open, titled 'Commit new file'. It contains fields for 'Initial commit' and 'Add an optional extended description...'. At the bottom are 'Commit new file' and 'Cancel' buttons. The GitHub interface includes navigation links like 'Code', 'Issues', 'Pull requests', 'Actions', 'Projects', 'Security', 'Insights', and 'Settings' at the top.

Fig. Creating member read only repositories

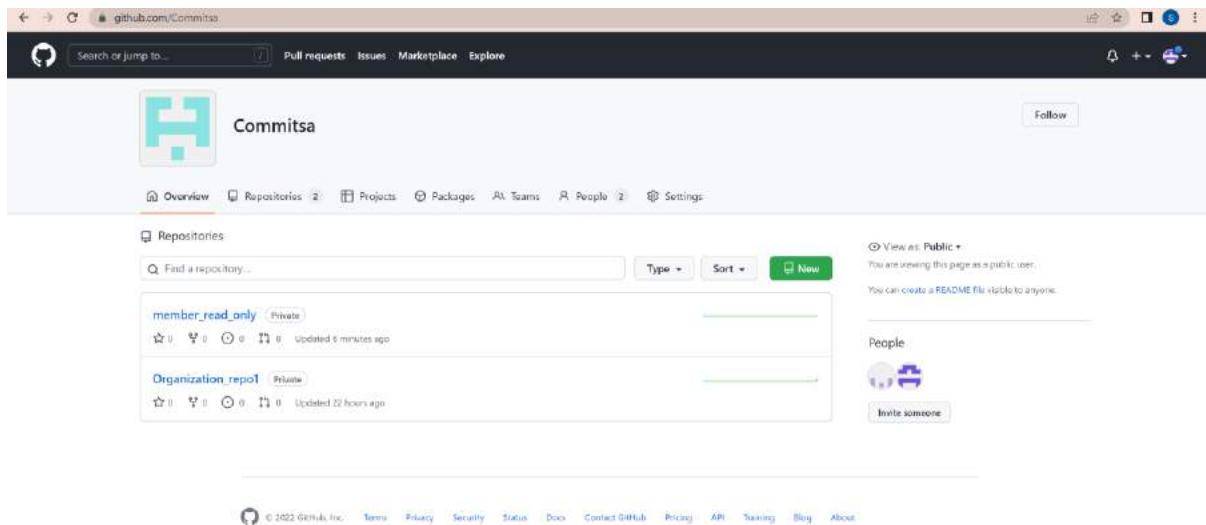


Fig. Successful Created Account

## Checking Access for Individual Repositories:

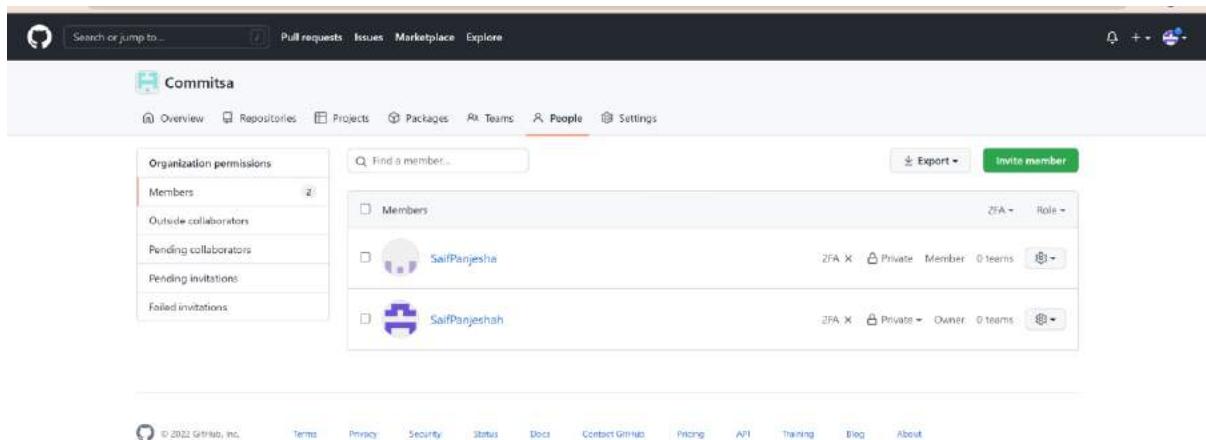


Fig. Checking access

The screenshot shows the GitHub organization settings for 'Commitsa'. On the left, there's a sidebar with a user profile for 'SaiPanjsha' (Role: Member) and stats: 2 repositories, 0 teams, Membership private, and Two-factor security disabled. Below this are 'Convert to outside collaborator' and 'Remove from organization' buttons. The main area displays repository access for 'SaiPanjsha' to two repositories: 'Commitsa/Organization\_repo1' and 'Commitsa/member\_read\_only'. Both have 'Write on the repository' permission, with 'Manage access' buttons. A search bar at the top right says 'Find a repository they have access to...'. At the bottom, there's a navigation bar with links like Overview, Repositories, Projects, Packages, AI Teams, People, and Settings.

Fig. Shows both the account has same access for repositories

The screenshot shows the GitHub organization settings for 'Commitsa'. The left sidebar includes sections for General, Access (Billing and plans, Member privileges selected), Write, and Pages. A modal window titled 'Change base permission to "Read"' is open, asking if the user wants to change the base repository permission for the organization. It explains that this will change the permission for all 2 repositories. Below the modal, the 'Organization member permissions' section is visible, showing 'No permission' (disabled), 'Read' (selected), and 'Write' (disabled). The 'Write' option is described as allowing members to clone and push to repositories. The 'Read' option is described as allowing members to clone and pull repositories. The 'Pages creation' section below is also visible.

Fig. Shows Permission may change for both the repositories

How to resolve this problem access Permission?

## By Introducing Teams in Organizational Account:

The screenshot shows the GitHub interface for an organizational account named 'Commitsa'. The top navigation bar includes 'Search or jump to...', 'Pull requests', 'Issues', 'Marketplace', 'Explore', and a user icon. Below the bar, the repository name 'Commitsa' is displayed, followed by tabs for 'Overview', 'Repositories', 'Projects', 'Packages', 'All Teams' (which is selected), 'All People', and 'Settings'. A prominent section titled 'Seamless communication with teams' highlights three features: 'Flexible repository access' (with a brief description and a circular icon showing a gear and a plus sign), 'Request to join teams' (with a brief description and a circular icon showing a person with a question mark), and 'Team mentions' (with a brief description and a circular icon showing a speech bubble with a checkmark). At the bottom of this section are two buttons: 'New team' (green) and 'Learn more'.

Fig. Teams Ribbons of Organizational Account

The screenshot shows the 'Create new team' form on GitHub. The top navigation bar and repository details are identical to the previous screenshot. The main form has a title 'Create new team'. It contains fields for 'Team name' (set to 'Commitsa\_Read\_Write\_teams'), 'Description' (empty), 'Parent team' (empty), and 'Team visibility' (set to 'Visible'). Below these are detailed descriptions of each setting. At the bottom is a green 'Create team' button. The footer of the page includes links for 'Terms', 'Privacy', 'Security', 'Status', 'Doc', 'Contact GitHub', 'Pricing', 'API', 'Training', 'Blog', and 'About'.

Fig. Creating Teams in Organization

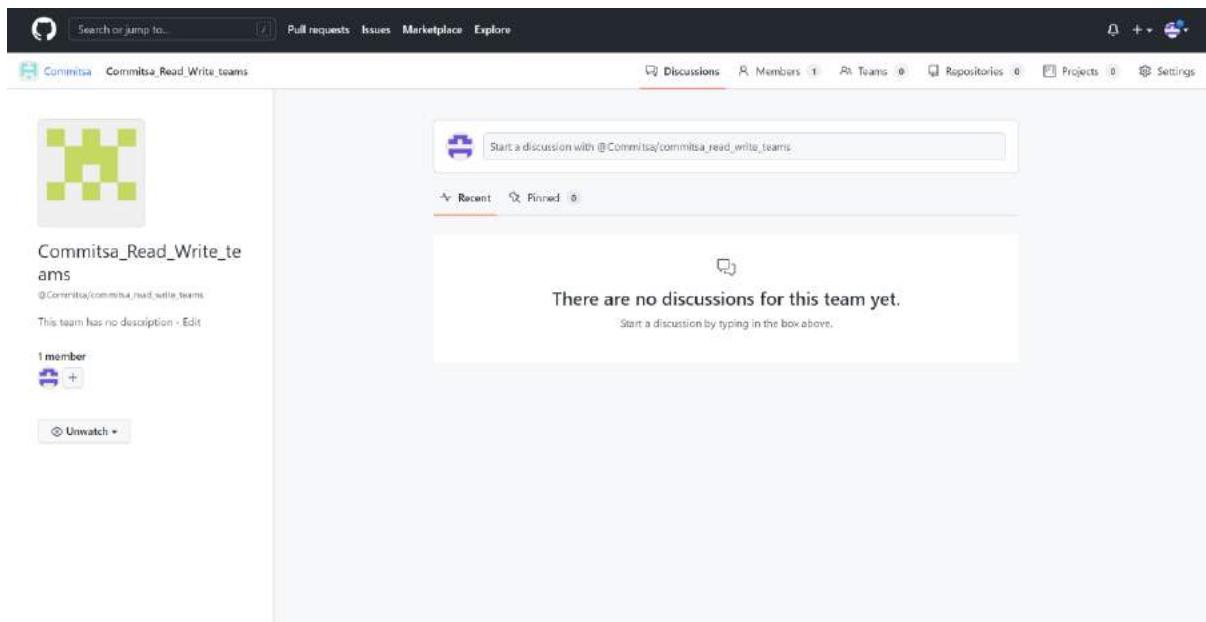


Fig. Read and Write team successfully Created

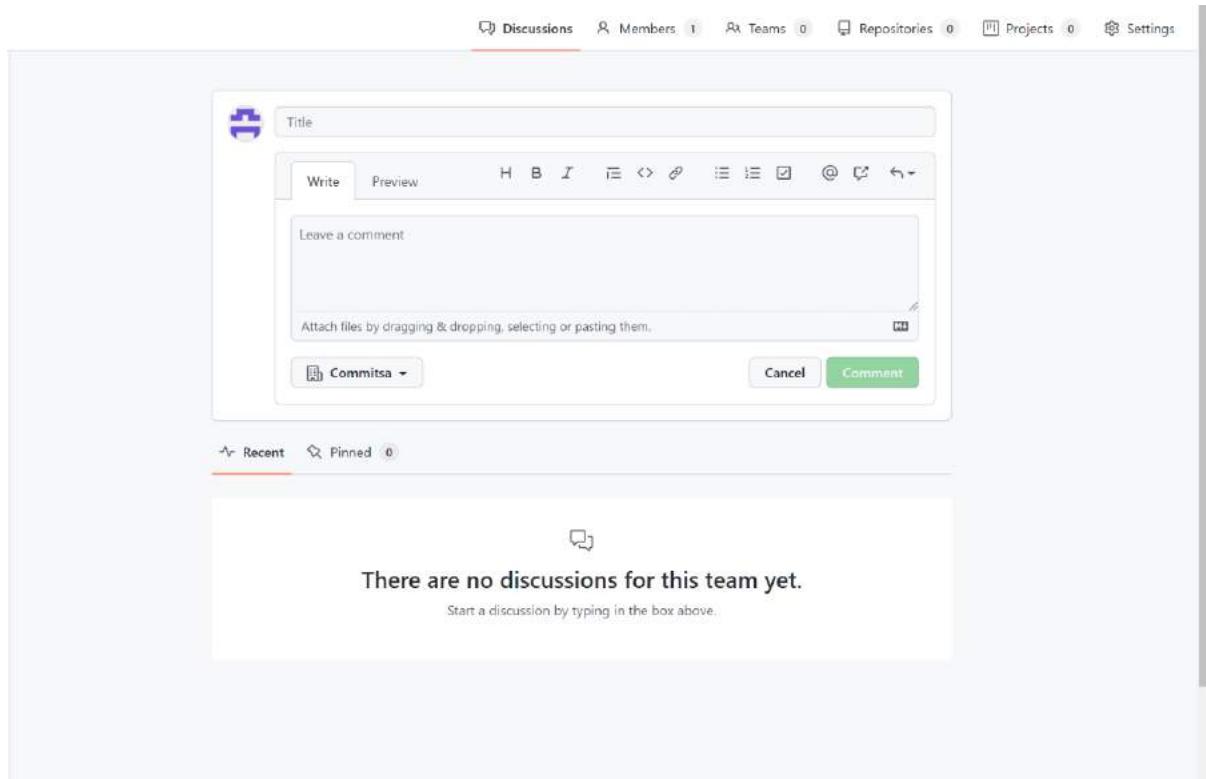


Fig. Forum of discussion ribbons in teams to check

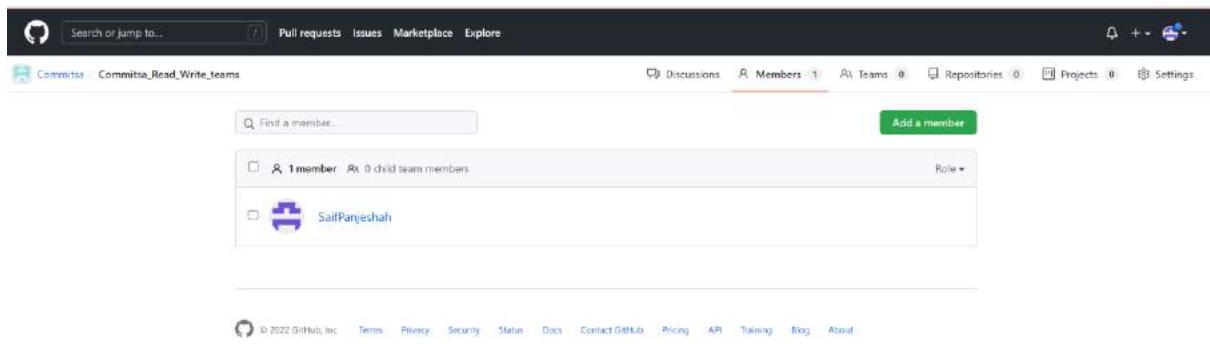


Fig. Owner as Team Member Default

## Let's create team repository:

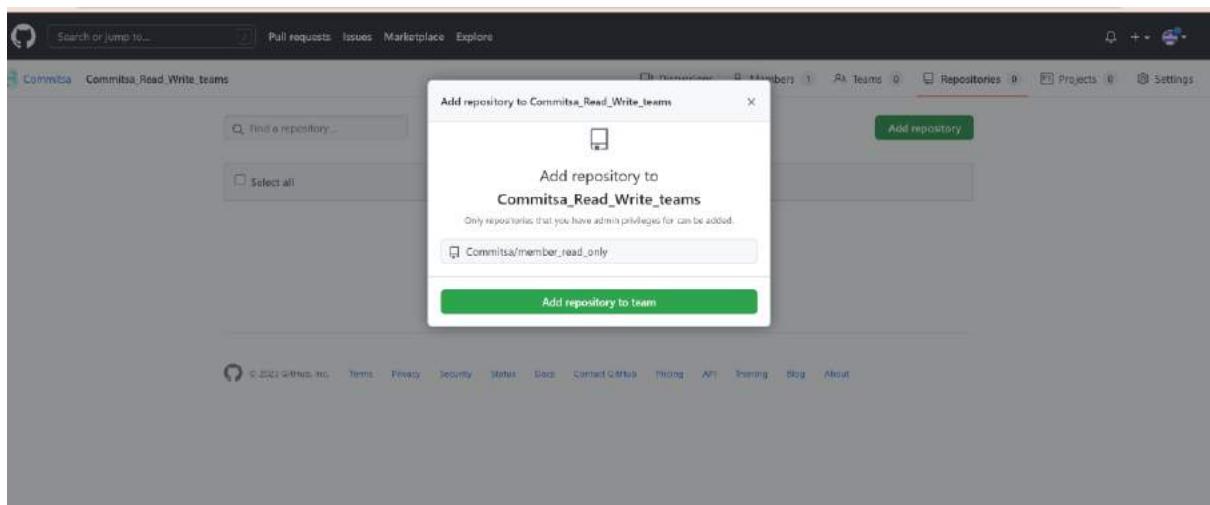


Fig. Creating Team Repository

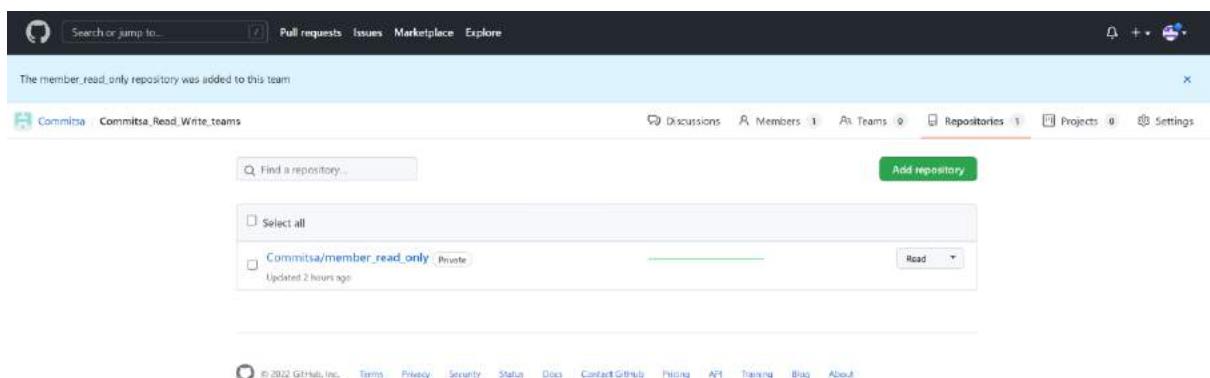


Fig. Repository Created with read access

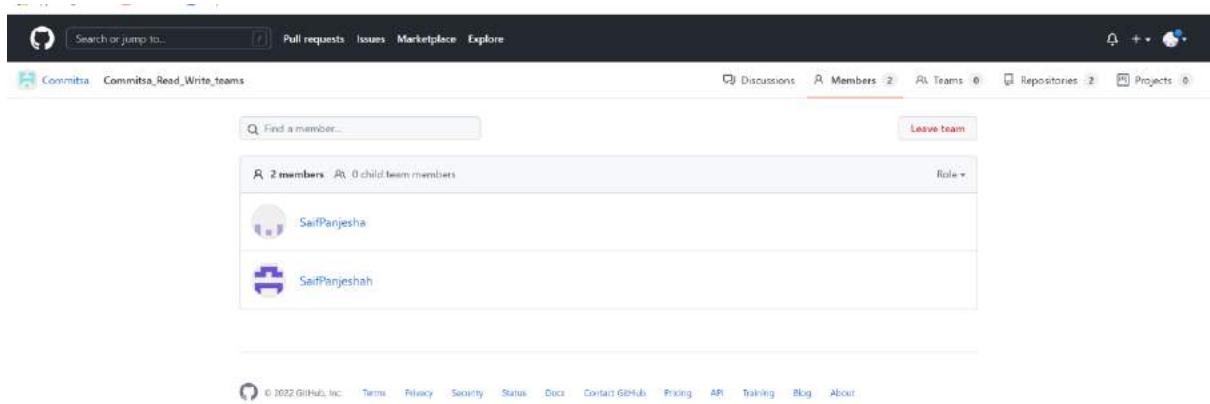


Fig. Successful adding team member

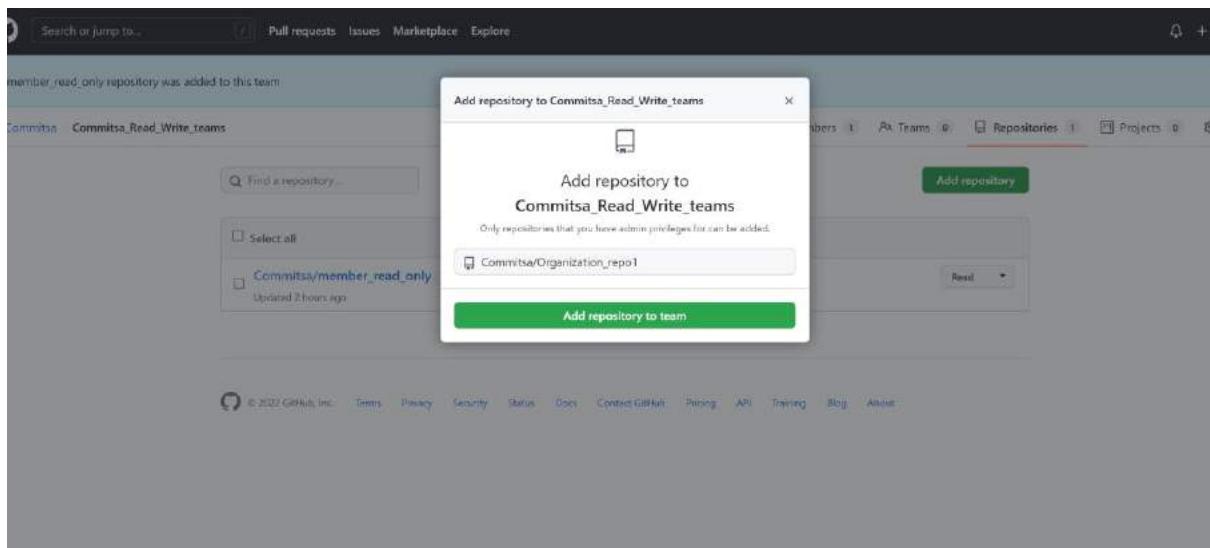


Fig. Creating Team Repository

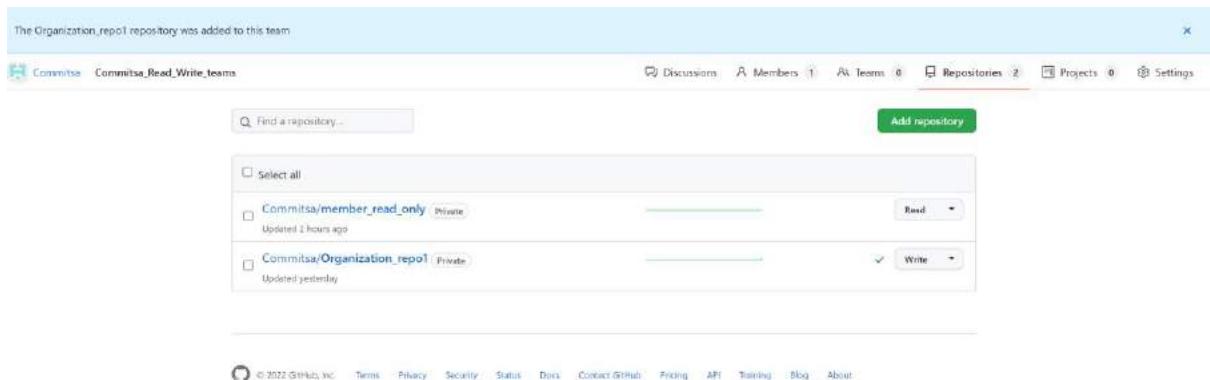
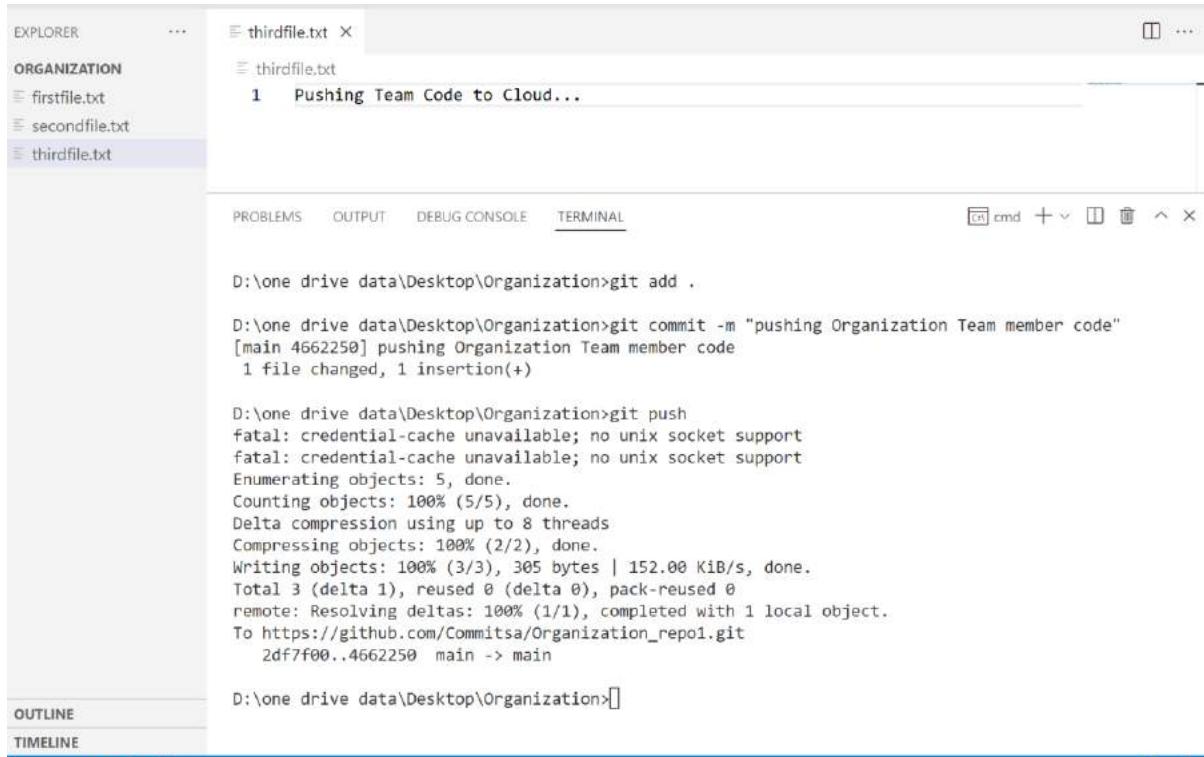


Fig. Successful Created repositories with write access

Hence, we can able to access for Individual Repositories with teams as flexible work.

### Managing Team Repositories Access Efficiently:

Let's Now, trying to push the code from local branch git (VS Code) to check whether the code is successful to cloud or not.



The screenshot shows the VS Code interface with the following details:

- EXPLORER**: Shows files: firstfile.txt, secondfile.txt, thirdfile.txt, and a folder named "thirdfile.txt".
- ORGANIZATION**: Shows files: firstfile.txt, secondfile.txt, and thirdfile.txt.
- TERMINAL**: Displays the following command-line session:

```
D:\one drive data\Desktop\Organization>git add .  
D:\one drive data\Desktop\Organization>git commit -m "pushing Organization Team member code"  
[main 4662250] pushing Organization Team member code  
1 file changed, 1 insertion(+)  
  
D:\one drive data\Desktop\Organization>git push  
fatal: credential-cache unavailable; no unix socket support  
fatal: credential-cache unavailable; no unix socket support  
Enumerating objects: 5, done.  
Counting objects: 100% (5/5), done.  
Delta compression using up to 8 threads  
Compressing objects: 100% (2/2), done.  
Writing objects: 100% (3/3), 305 bytes | 152.00 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.  
To https://github.com/Commitsta/Organization_repo1.git  
2df7f00..4662250 main -> main  
  
D:\one drive data\Desktop\Organization>
```
- OUTLINE** and **TIMELINE** are also visible on the left.

Fig. Successful Push coding on Organization repository with write access

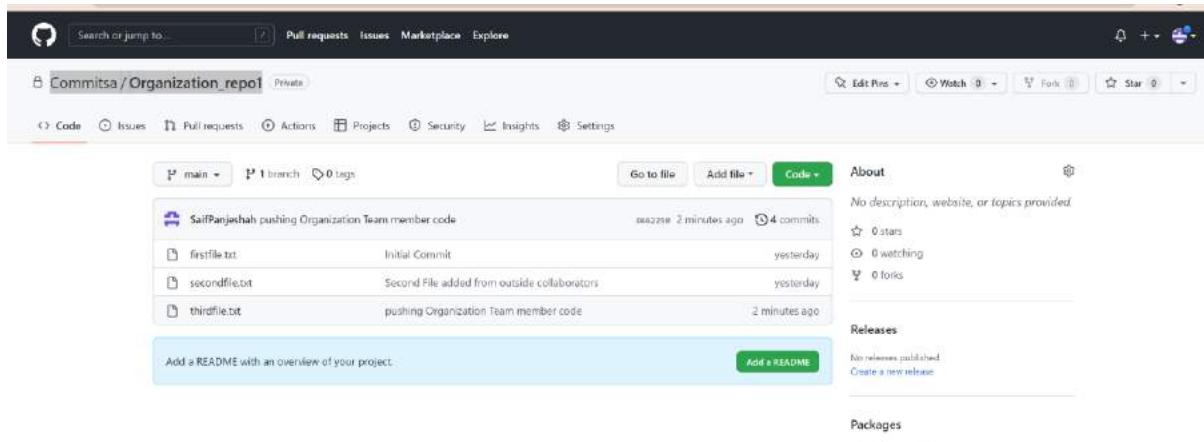


Fig. Successful Push on GitHub

**Let's Push our code on: [https://github.com/Commitsa/member\\_read\\_only.git](https://github.com/Commitsa/member_read_only.git).**

The screenshot shows a Visual Studio Code interface with the following details:

- EXPLORER**: Shows a folder named "ORGANIZATIONS" containing a file "read.txt". The content of "read.txt" is:

```
1 This is the First Text!
2 Let's Push Code on Cloud !!!
```
- ORGANIZATIONS**: A section showing the file "read.txt" with its content.
- TERMINAL**: An open terminal window with the following command history:

```
Microsoft Windows [Version 10.0.22000.675]
(c) Microsoft Corporation. All rights reserved.

D:\one drive data\Desktop\Organizations>git clone https://github.com/Commitssa/member_read_only.git .
Cloning into '.'...
fatal: credential-cache unavailable; no unix socket support
fatal: credential-cache unavailable; no unix socket support
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.

D:\one drive data\Desktop\Organizations>git add .

D:\one drive data\Desktop\Organizations>git commit -m " Succesful Push Code on Cloud .."
[main 696f26a] Succesful Push Code on Cloud ..
1 file changed, 1 insertion(+)

D:\one drive data\Desktop\Organizations>git push
fatal: credential-cache unavailable; no unix socket support
fatal: credential-cache unavailable; no unix socket support
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 314 bytes | 314.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Commitssa/member_read_only.git
 f5d1833..696f26a main -> main
```
- PROBLEMS**, **OUTPUT**, **DEBUG CONSOLE**: Other tabs in the terminal bar.
- cmd**: A button in the top right corner of the terminal.
- OUTLINE** and **TIMELINE**: Buttons at the bottom left.

Fig. Successful Push coding on member\_read\_only repository with read access

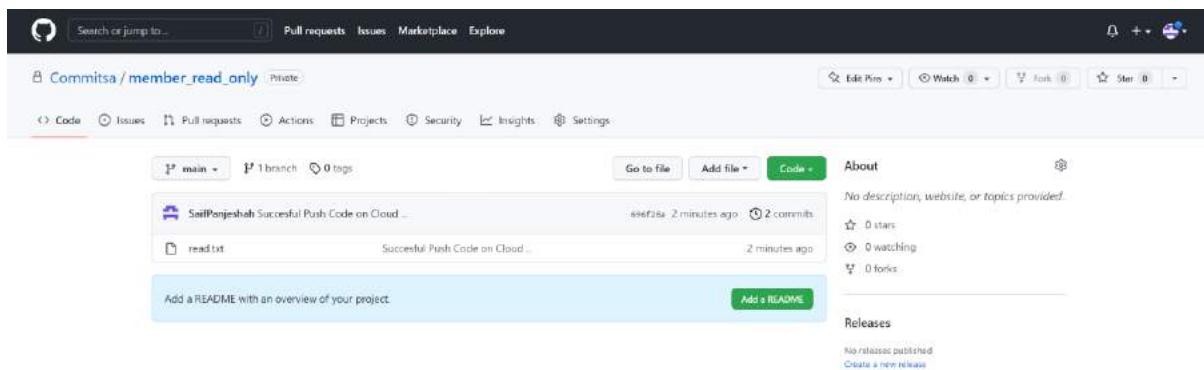


Fig. Successful Push on GitHub

## **Understanding Forks and Pull Request:**

**Definition: A fork is a copy of a repository that you manage. Forks let you make changes to a project without affecting the original repository. You can fetch updates from or submit changes to the original repository with pull requests.**

Fork	Clone
Forking is done on the GitHub Account	Cloning is done using Git
Forking a repository creates a copy of the original repository on our GitHub account	Cloning a repository creates a copy of the original repository on our local machine
Changes made to the forked repository can be merged with the original repository via a pull request	Changes made to the cloned repository cannot be merged with the original repository unless you are the collaborator or the owner of the repository
Forking is a concept	Cloning is a process
Forking is just containing a separate copy of the repository and there is no command involved	Cloning is done through the command ' <b>git clone</b> ' and it is a process of receiving all the code files to the local machine

**Fig. Difference between Fork & Clone**

**When to use:** A fork is a rough copy of a repository. Forking a repository allows you to freely test and debug with changes without affecting the original project. One of the excessive uses of forking is to propose changes for bug fixing. To resolve an issue for a bug that you found, you can:

- Fork the repository.
- Make the fix.
- Forward a pull request to the project owner

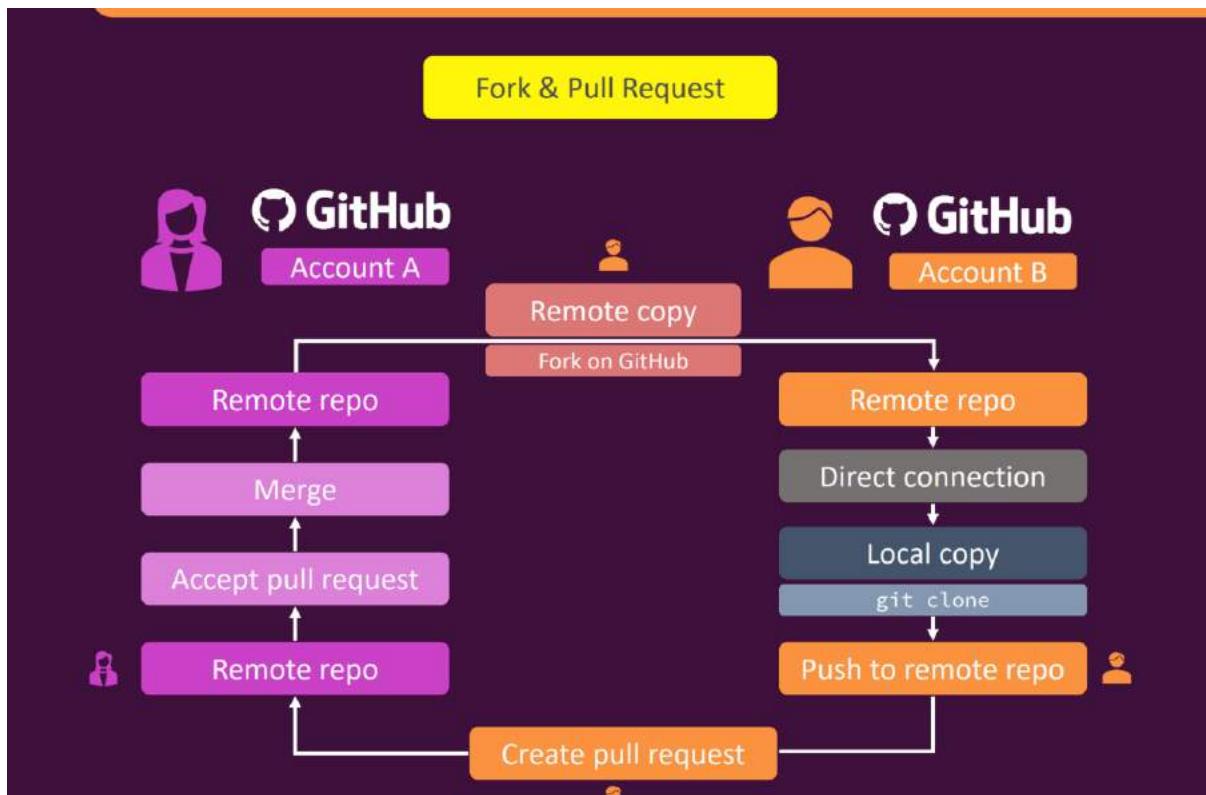


Fig. Creating Fork & Pull Request

## 1. Forking the Repository:

Assuming you're using GitHub, this step is easy. Just find the repository you're contributing to and press the Fork button in the upper right. This will create an exact copy of the repository (and all of its branches) under your own username

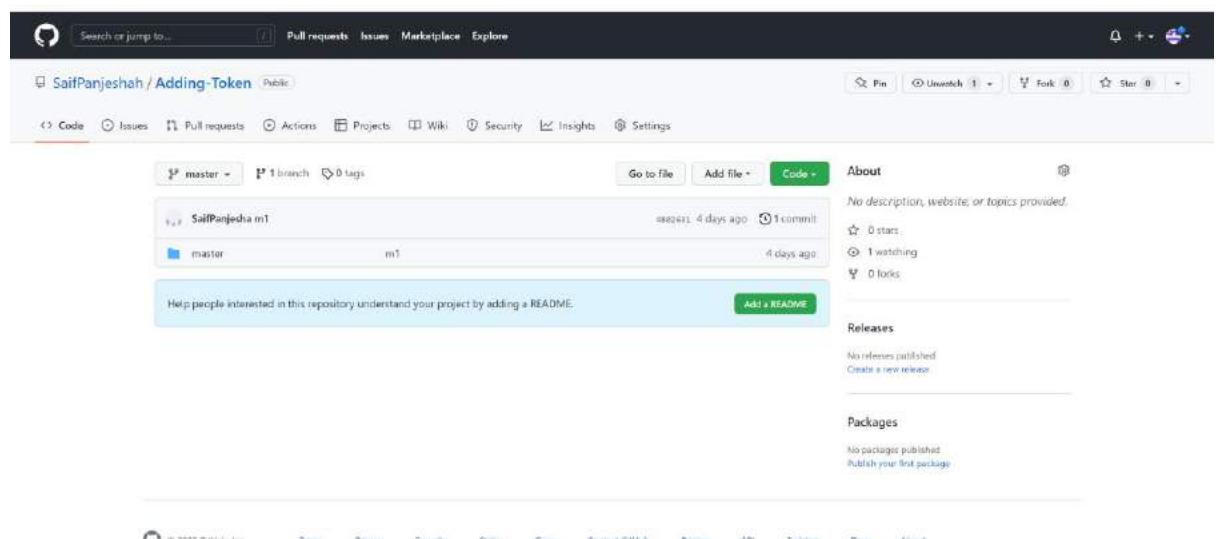


Fig. Forking the repository from another account

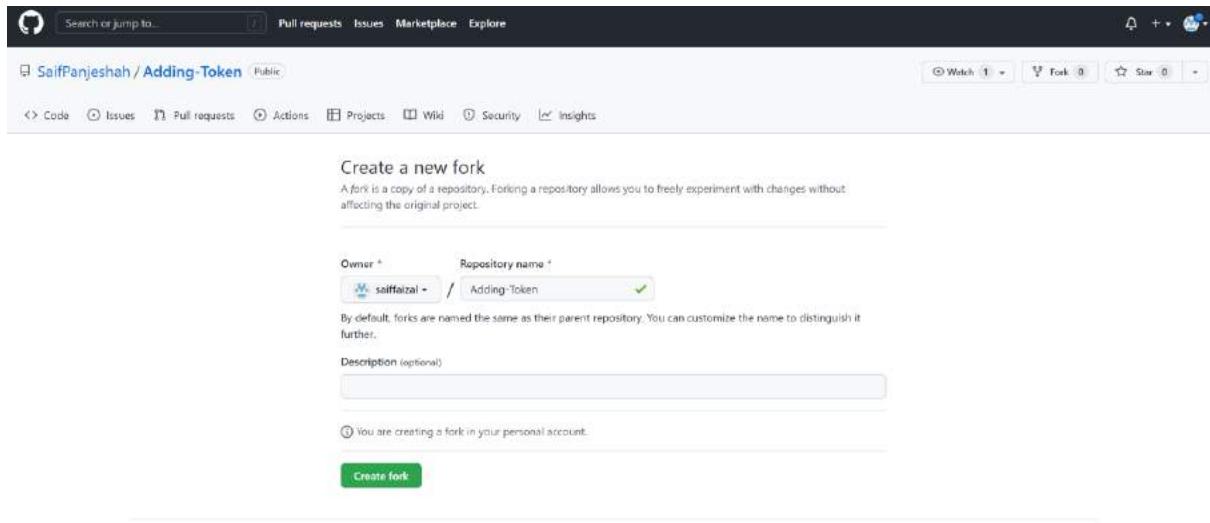


Fig. Create a new fork

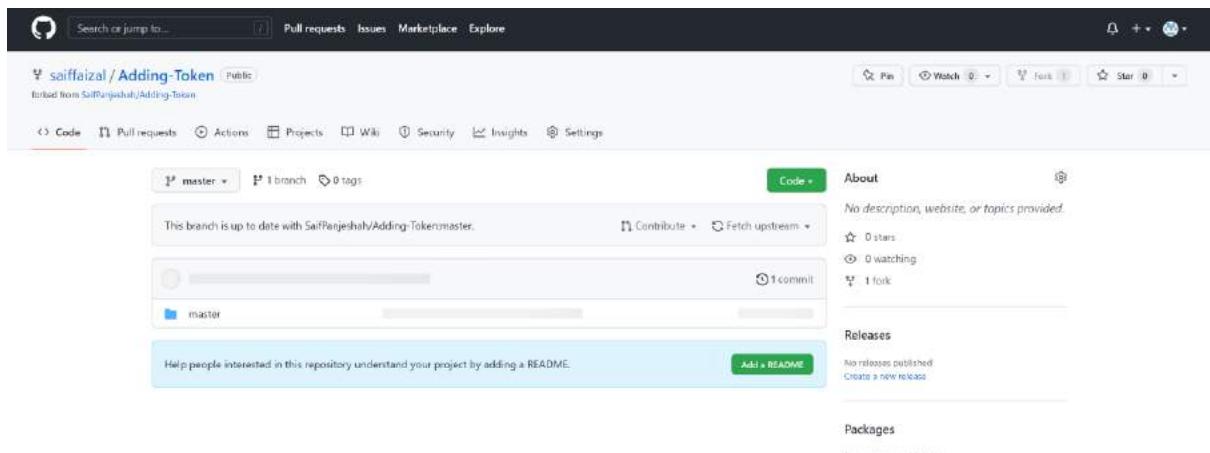


Fig. Forked Successful

## 2. Clone your new fork locally

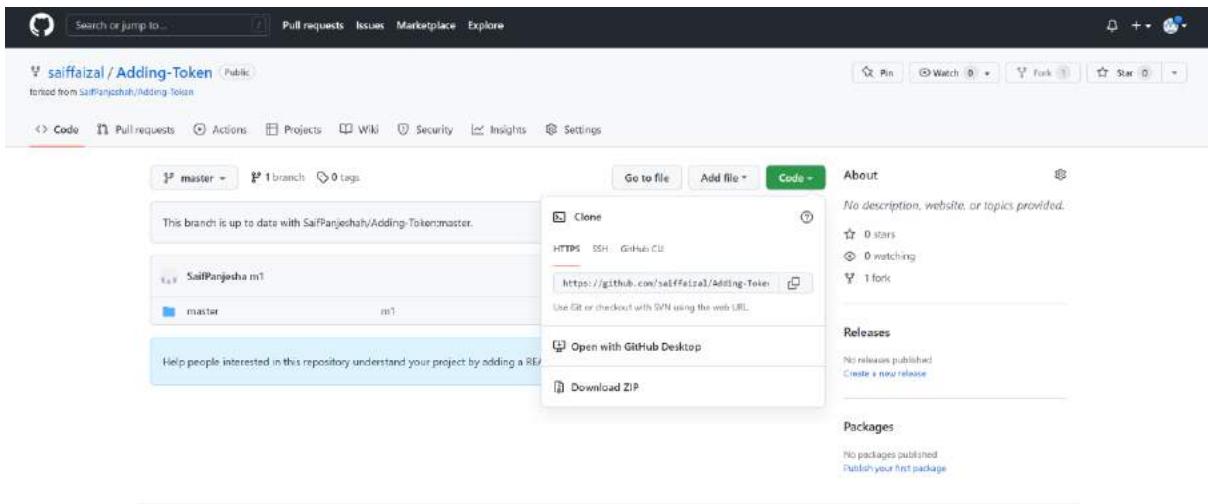


Fig. cloning the repository.

**Let's Now, trying to push the code from local branch VS code console to check whether the code is successful to cloud or not.**

```
D:\one drive data\Desktop\Forking Repositories>git clone https://github.com/saiffaizal/Adding-Token.git .
Cloning into '.'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 4 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.

D:\one drive data\Desktop\Forking Repositories>git add .

D:\one drive data\Desktop\Forking Repositories>git commit -m "Fork added"
[master 7444ed8] Fork added
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Fork/addingnewFeature2.txt

D:\one drive data\Desktop\Forking Repositories>git push
fatal: credential-cache unavailable; no unix socket support
fatal: credential-cache unavailable; no unix socket support
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 330 bytes | 330.00 KiB/s, done.
Total 4 (delta 0), reused 1 (delta 0), pack-reused 0
To https://github.com/saiffaizal/Adding-Token.git
 4802631..7444ed8 master -> master
```

Fig. Success to push the code

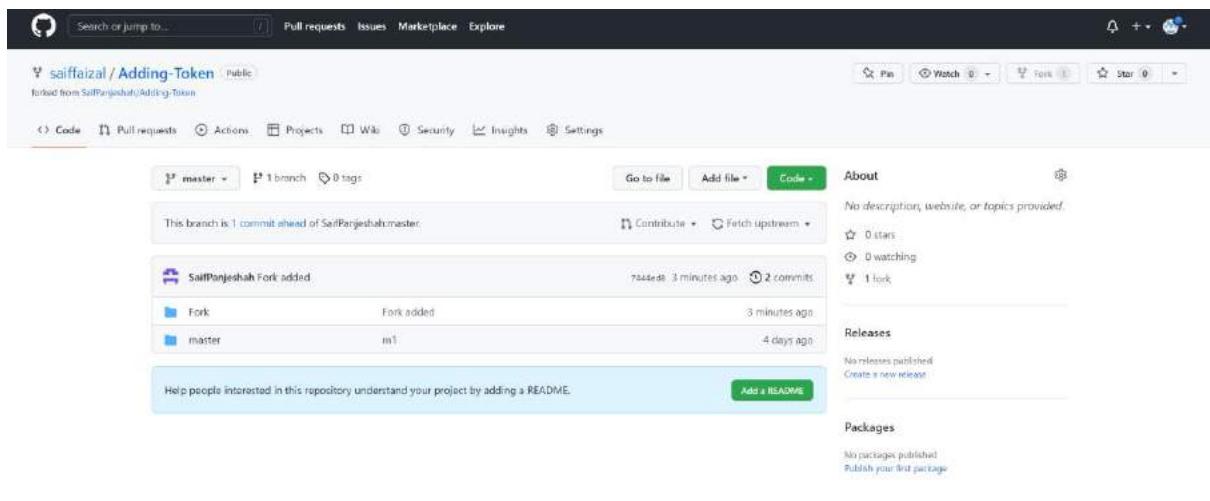


Fig. Successful on GitHub to member account

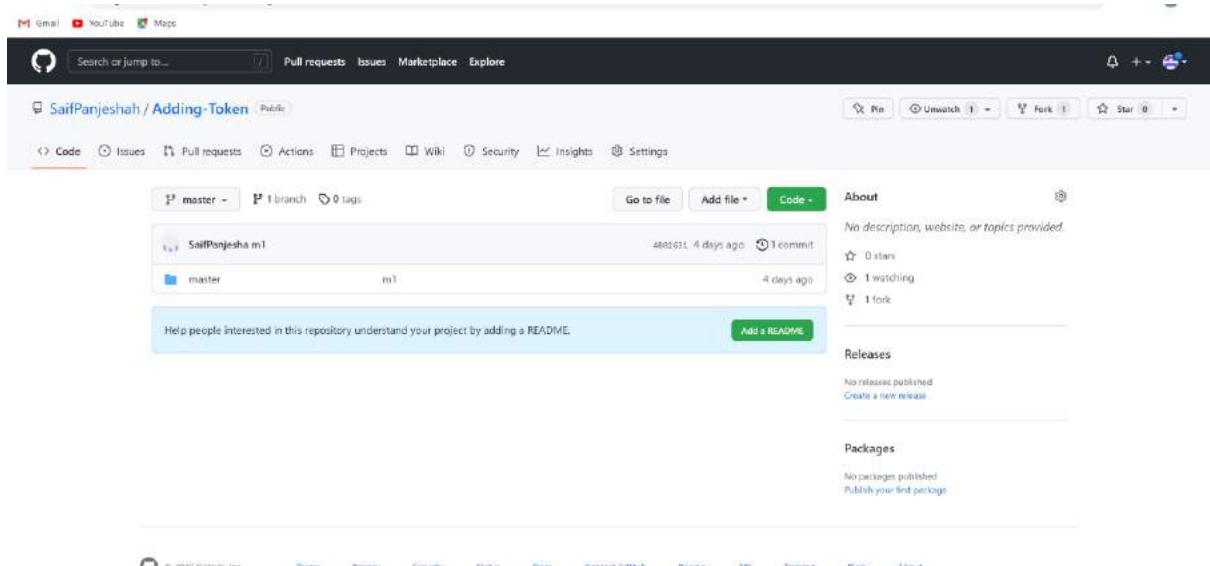


Fig. Not Fork Update Successful on GitHub owner account

How To resolve this problem?

## By using pull requests in practise:

### Creating a pull request

1. Switch to the branch that you want to create a pull request for [Saiffaizal/Adding-Token](#)
2. Click **Create Pull Request**. GitHub Desktop will open your default browser to take you to GitHub

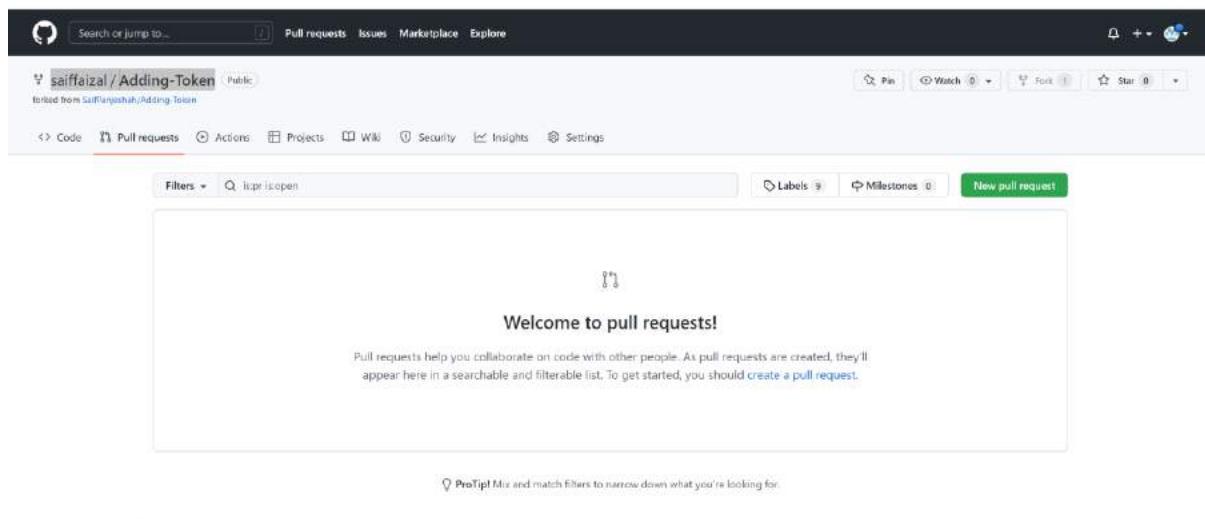


Fig. Creating a pull request

3. On GitHub, confirm that the branch in the **base**: drop-down menu is the branch where you want to merge your changes. Confirm that the branch in the **compare**: drop-down menu is the topic branch where you made your changes.

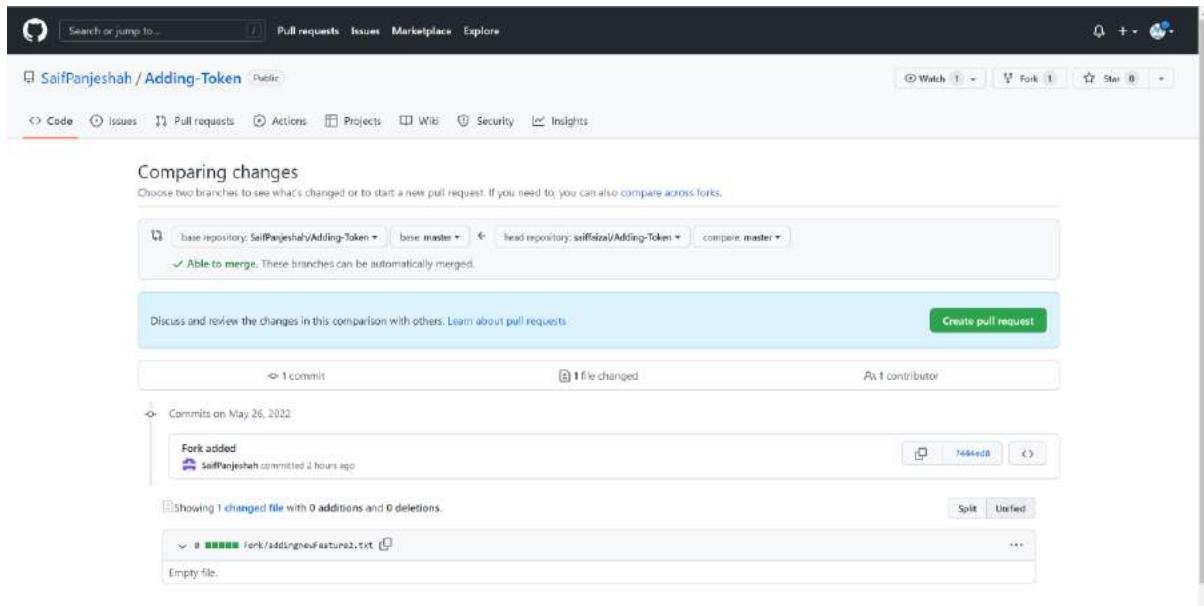


Fig. Comparing Changes

#### 4. Type a title and description for your pull request.

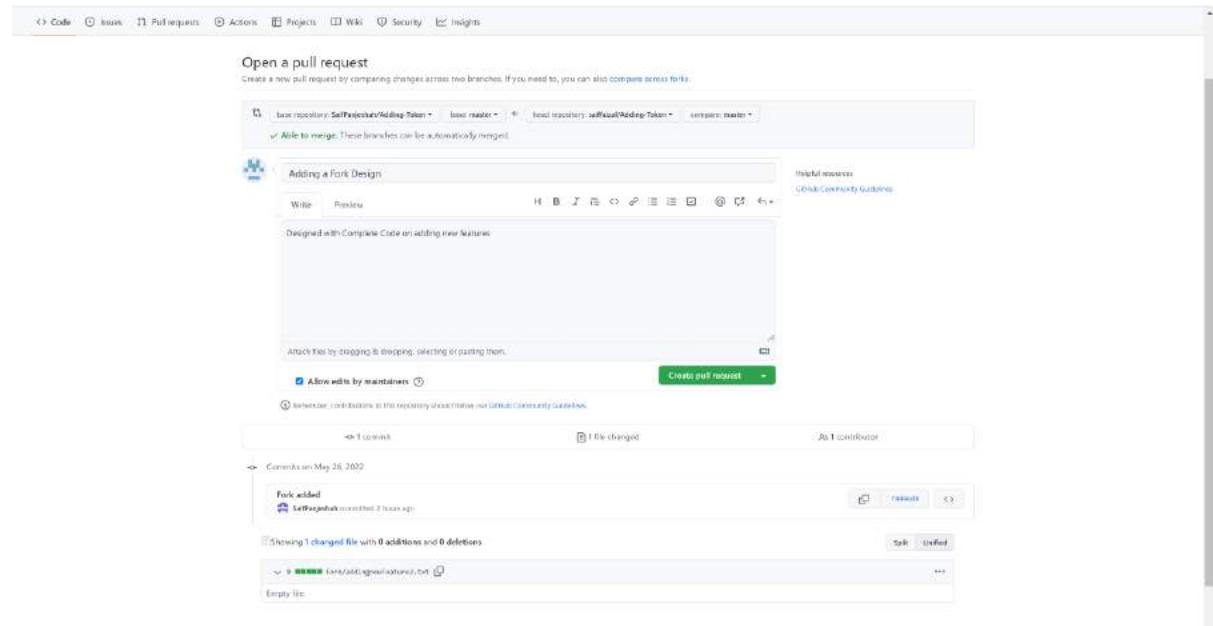


Fig. open pull request

## 5. To create a pull request that is ready for review, click Create Pull Request.

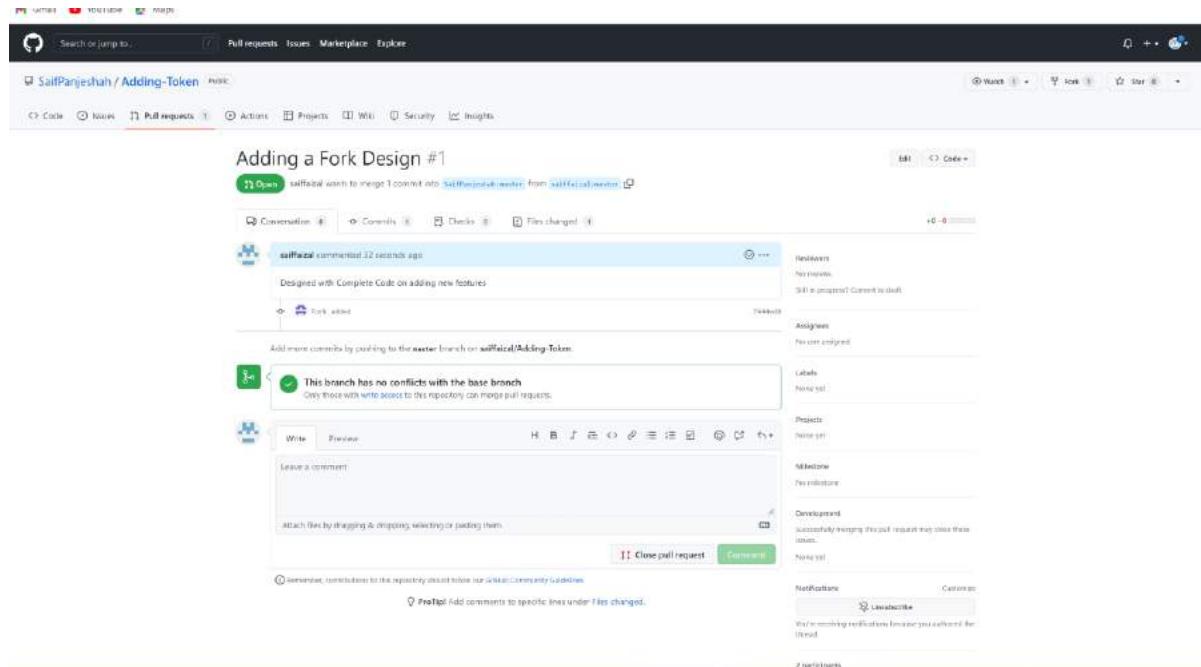


Fig. open pull request successful

Now, to view this pull request open owner organizational account

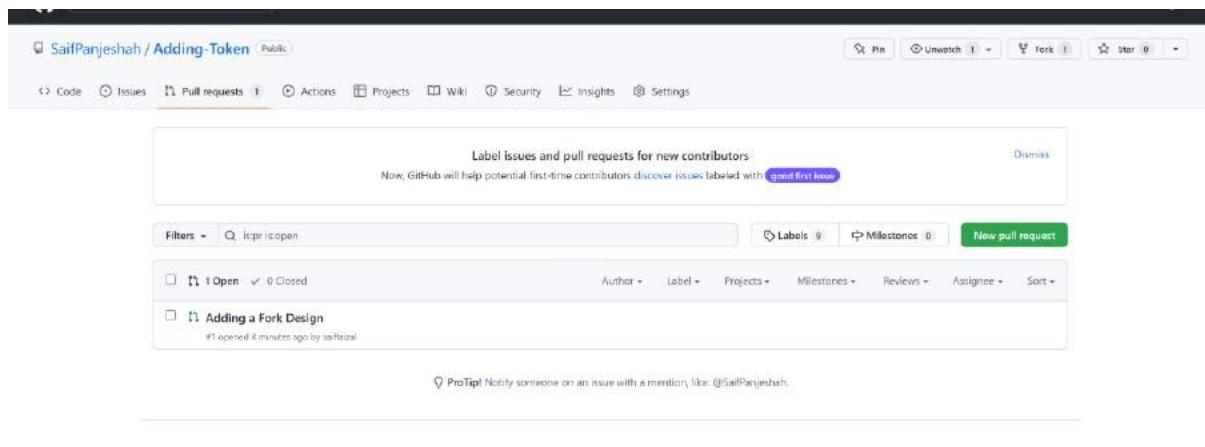


Fig. Owner view the pull request

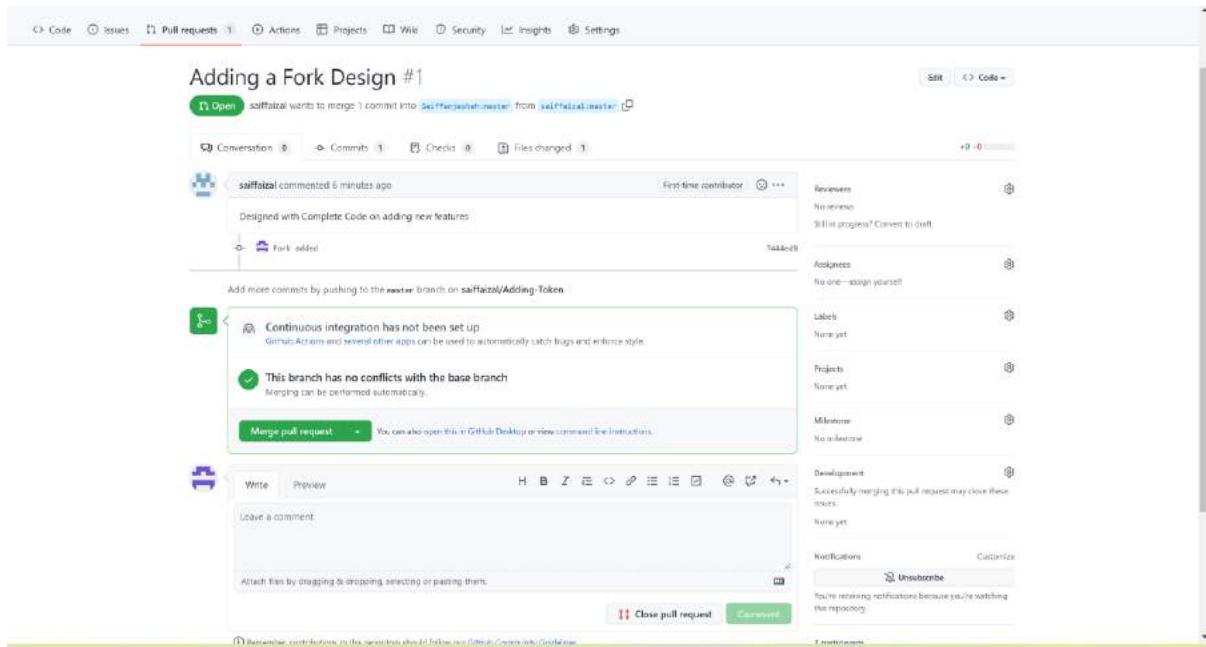


Fig. Merging or closing pull request

**If owner don't like and don't want to merge this pull request simply close this pull request**

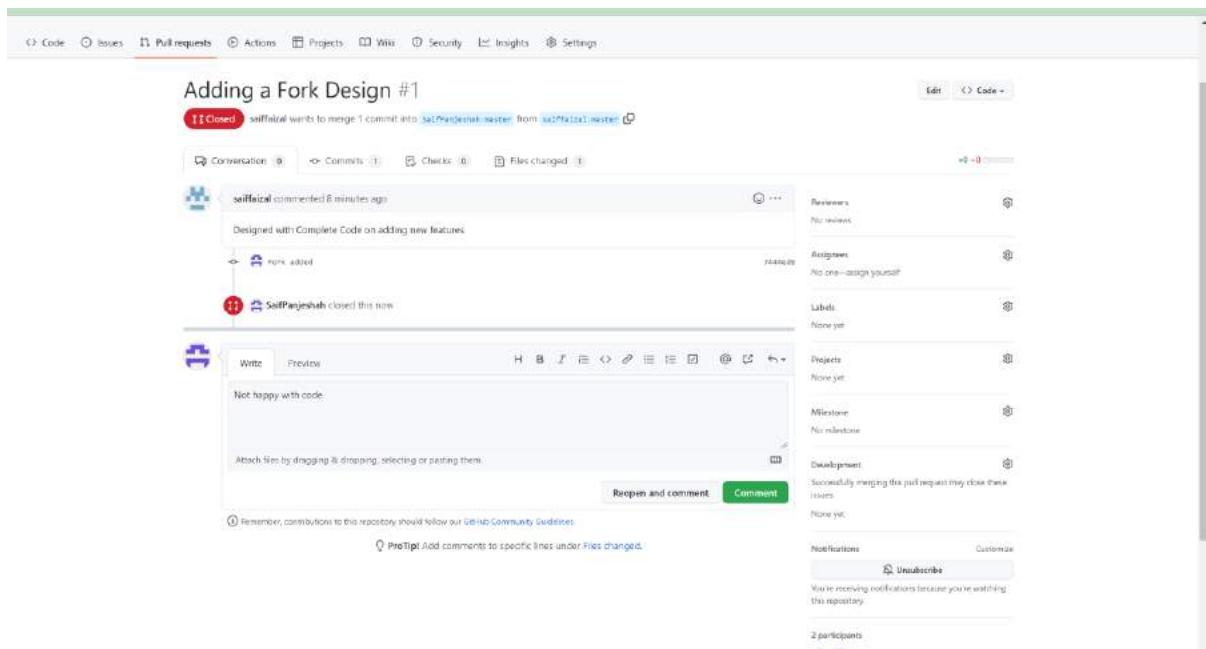


Fig. Owner closing this pull request

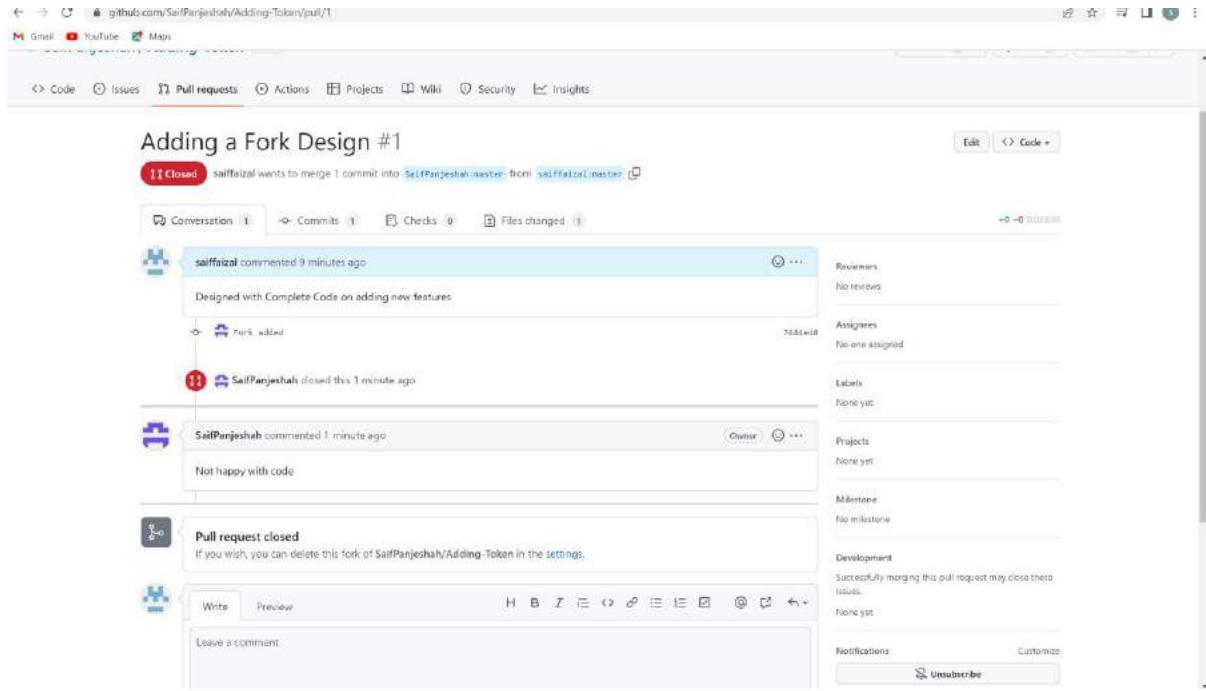


Fig. Closed Successful view from Member account

**Now, Owner Thinks a Second Way like the Fork design and wants to merge the code**

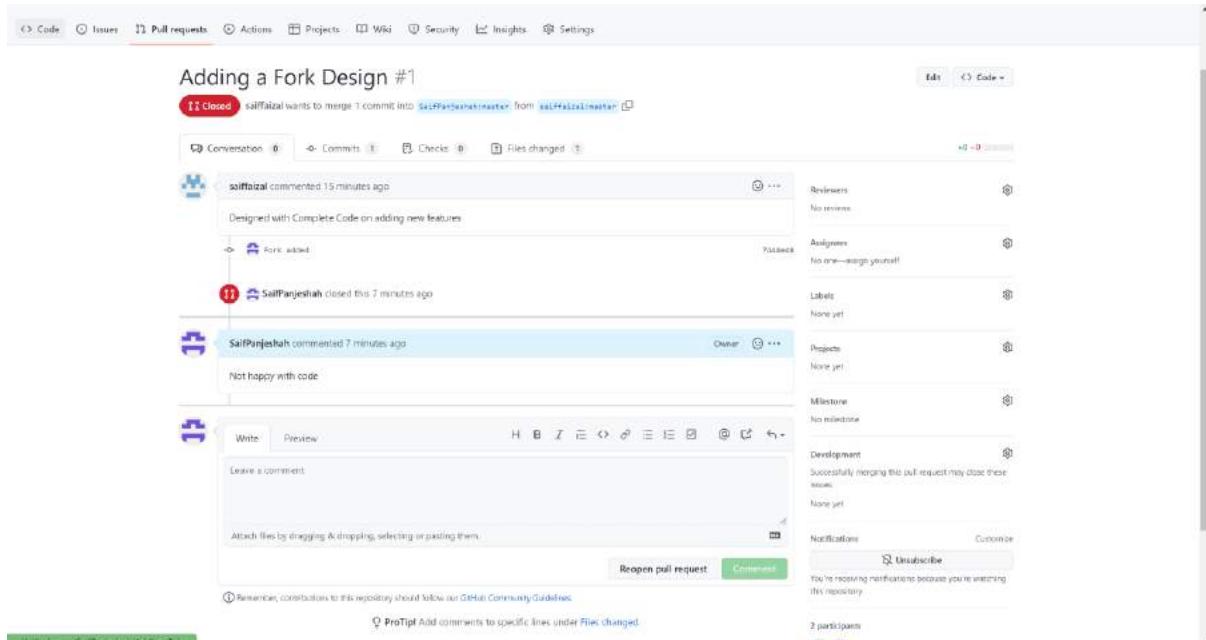


Fig. reopen the pull requests

## Merging the pull requests:

The screenshot shows a GitHub pull request interface for a repository named 'Adding a Fork Design'. The pull request has been opened by 'saifalzai' and is currently being merged into the 'master' branch from the 'saifalzai' fork. A comment from 'SaifPanjehshah' states 'Not happy with code'. The pull request details show a conflict: 'Continuous integration has not been set up' and 'This branch has no conflicts with the base branch'. A 'Merge pull request' button is present. The right sidebar displays repository metadata: Labels (None yet), Project (None yet), Milestone (No milestone), Development (Successfully merging this pull request may close these issues: None yet), Notifications (Customize, Unsubscribe), and Participants (2 participants).

Fig. Merging the pull requests

This screenshot shows the same GitHub pull request interface after the merge process. The pull request now has a status message indicating 'Merge pull request #1 from saifalzai/master' and 'Adding a Fork Design thanks saifalzai'. A 'Confirm merge' button is visible. The right sidebar remains the same, showing repository metadata and participant information.

Fig. Confirm merging

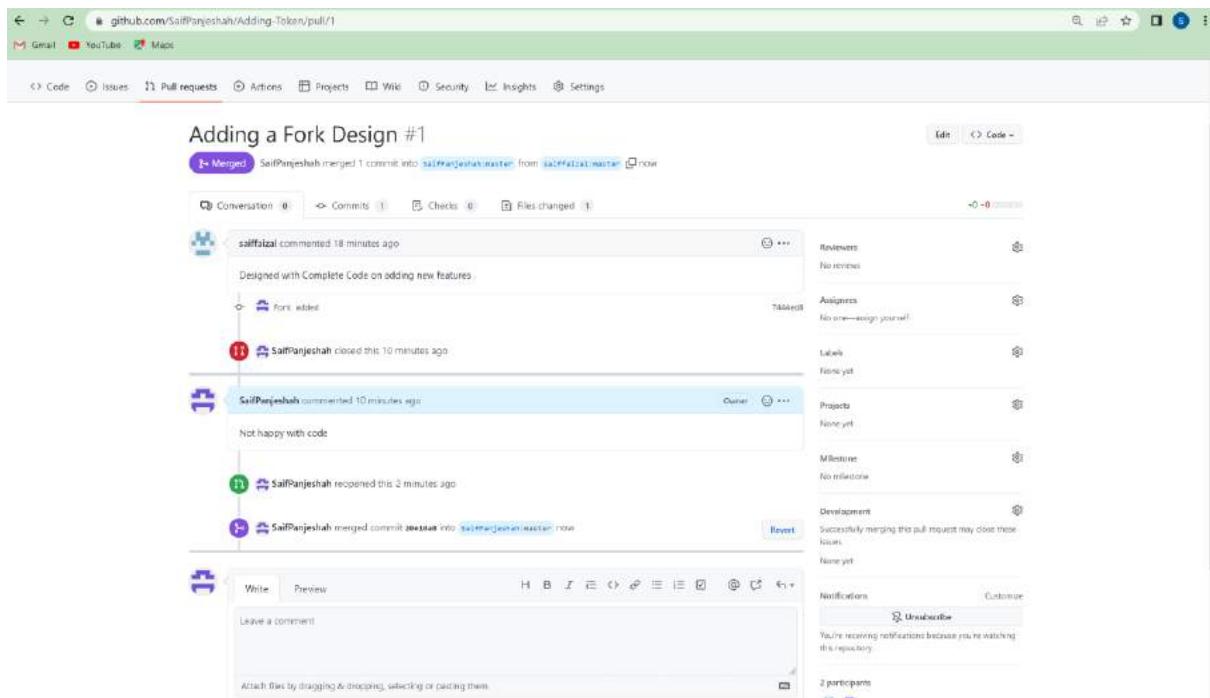


Fig. Merge successful

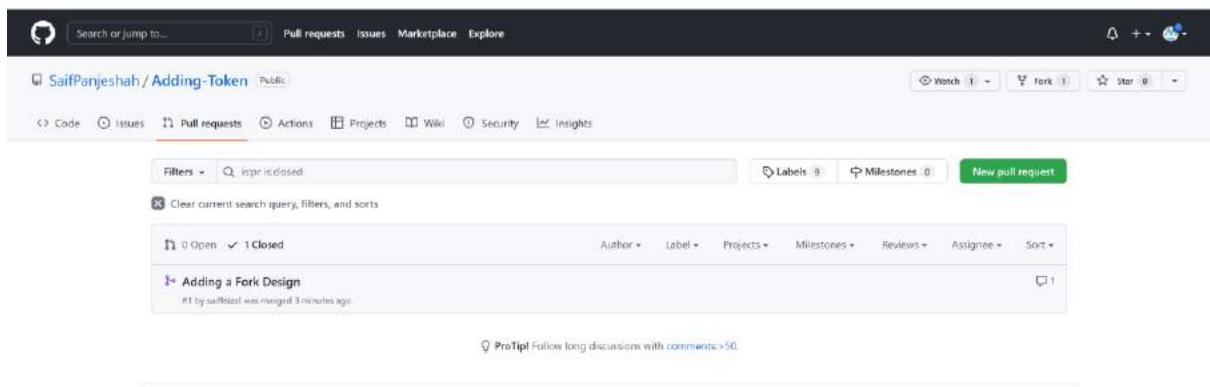


Fig. Close and successful Merge view from member account

## Openings & Closing Issues:

### Opening the Issue:

#### Creating an Issue from a repository:

1. On GitHub.com, navigate to the main page of the repository Under your repository name, click on Issues ribbon.

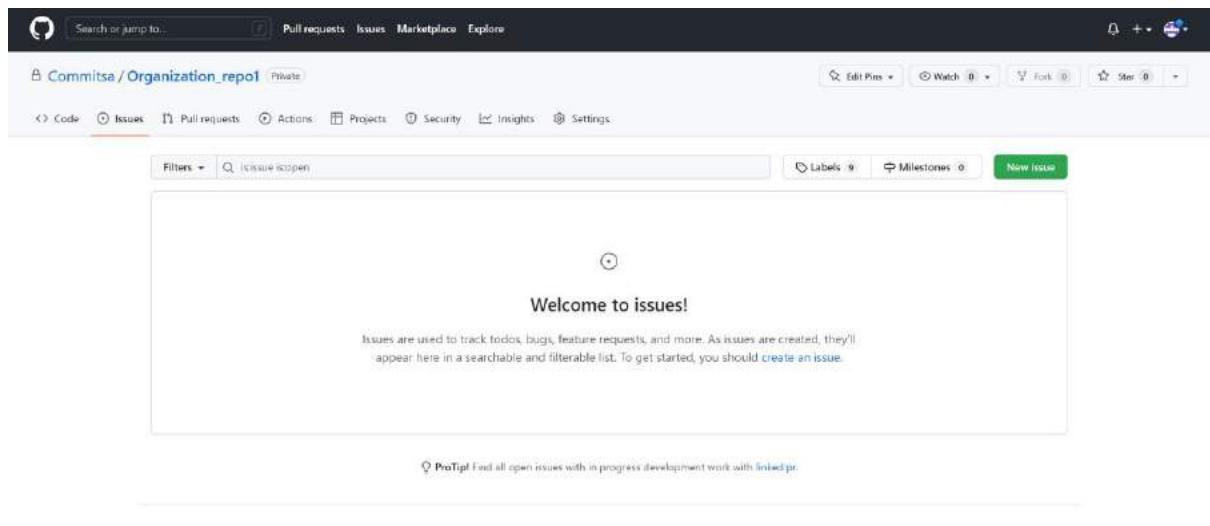


Fig. Creating a new Issue on organizational account repository

2. Click New issue:

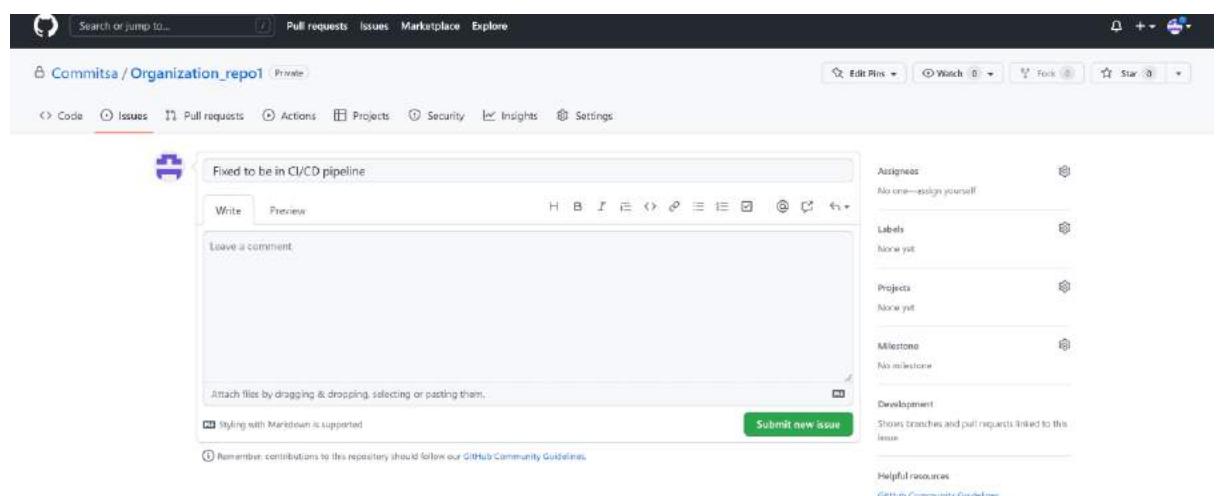


Fig. Creating Issues with CI/CD Pipelines of Devops

**Lets assigned this to member user of this organization and add labels as bugs needed to be fix:**

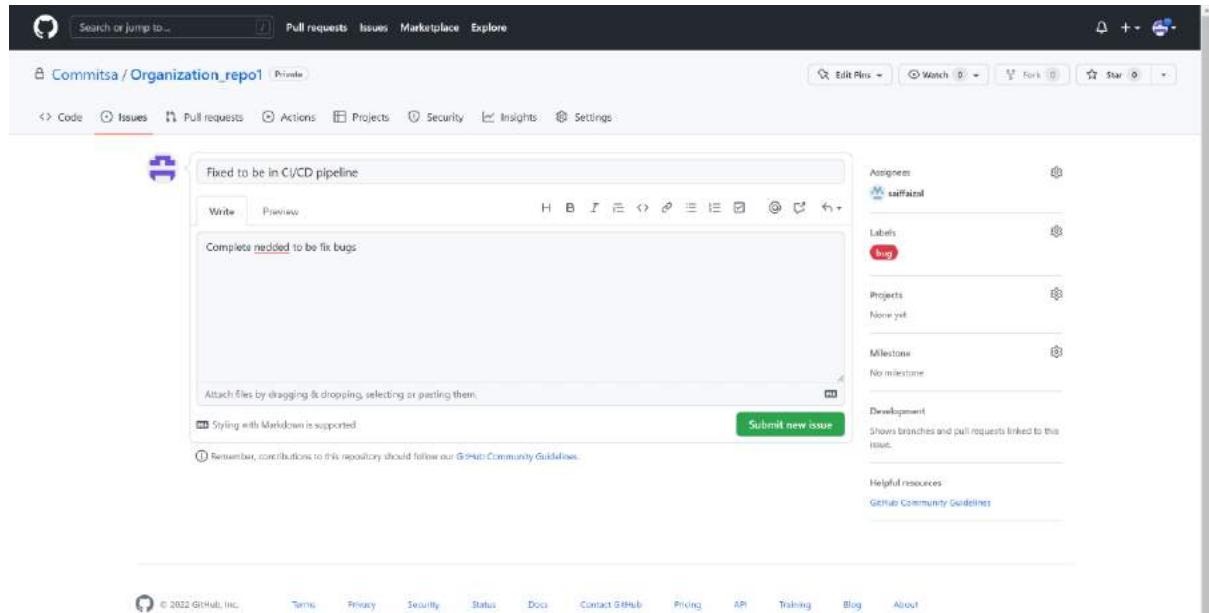


Fig. Fixed needs to be done

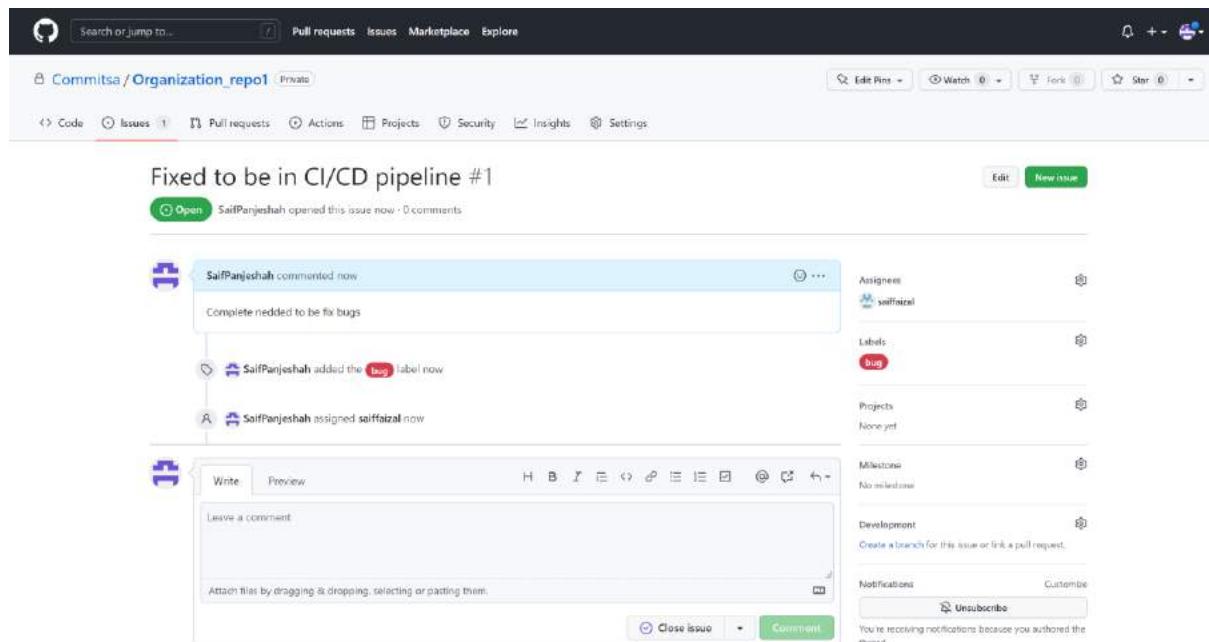


Fig. Issue successful Created

## Closing the Issue:

The screenshot shows a GitHub repository interface. At the top, there's a navigation bar with links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. Below the navigation is a search bar and a filter section. The main area displays an issue titled 'Fixed to be in CI/CD pipeline' with a 'bug' label. A note indicates it was opened 2 minutes ago by 'SaifPanjehah'. The issue has one comment from the same user, which includes a link to a commit. There are also sections for 'Assignees', 'Labels', 'Projects', 'Milestone', 'Development', 'Notifications', and 'Participants'.

Fig. view on member account

This screenshot shows the same GitHub issue page after the member has fixed it. The title now reads 'Fixed to be in CI/CD pipeline #1'. The member's comment has been updated to say 'I fixed this issues in my latest commit please check'. The 'Status' of the issue is now 'closed'. The rest of the interface remains the same, showing the assignee, labels, projects, milestones, development details, notifications, and participants.

Fig. Member Fixed this issue

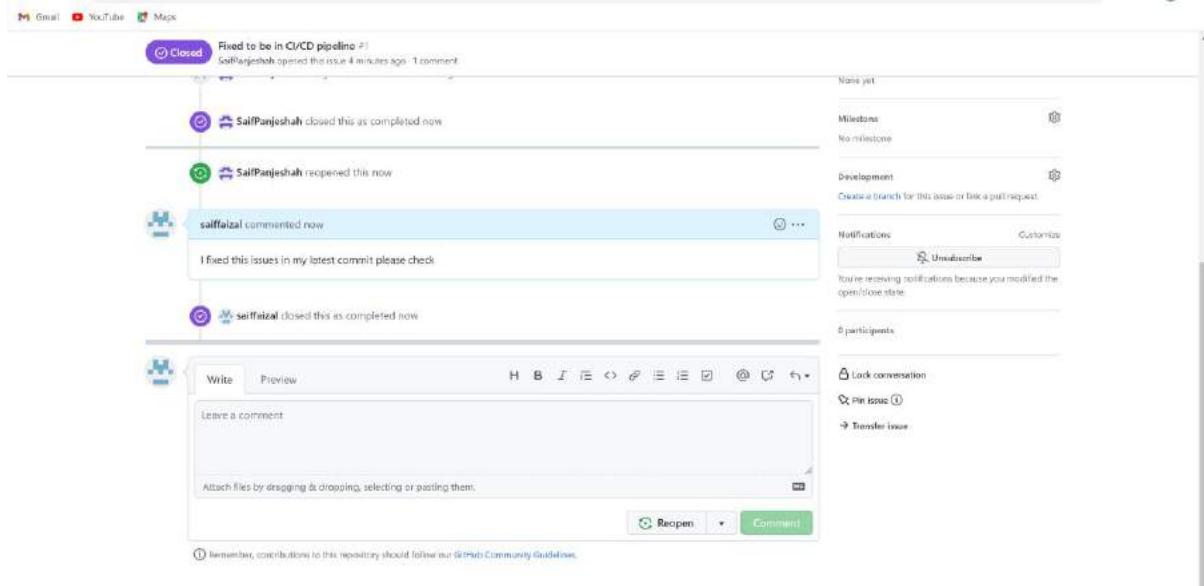


Fig. successful added close and fix commits

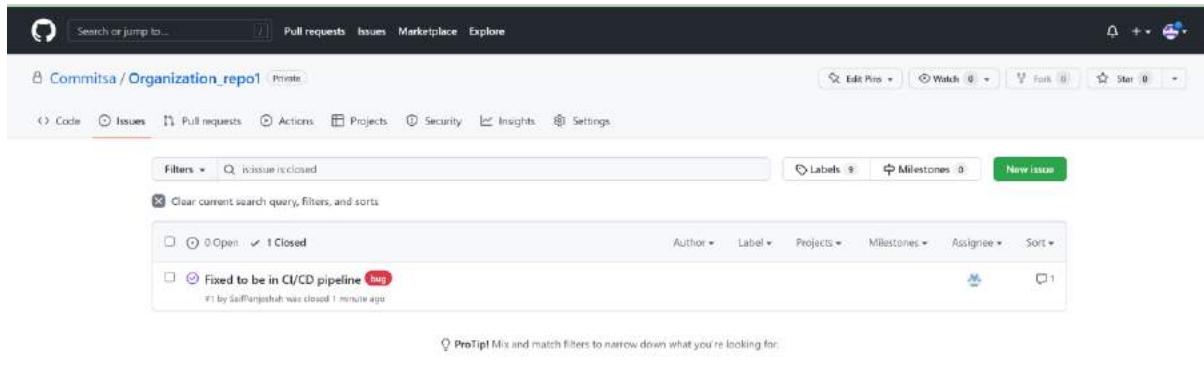


Fig. Successful close this Issue

## Working with GitHub Projects:

Let's Explore the Project ribbon in GitHub:

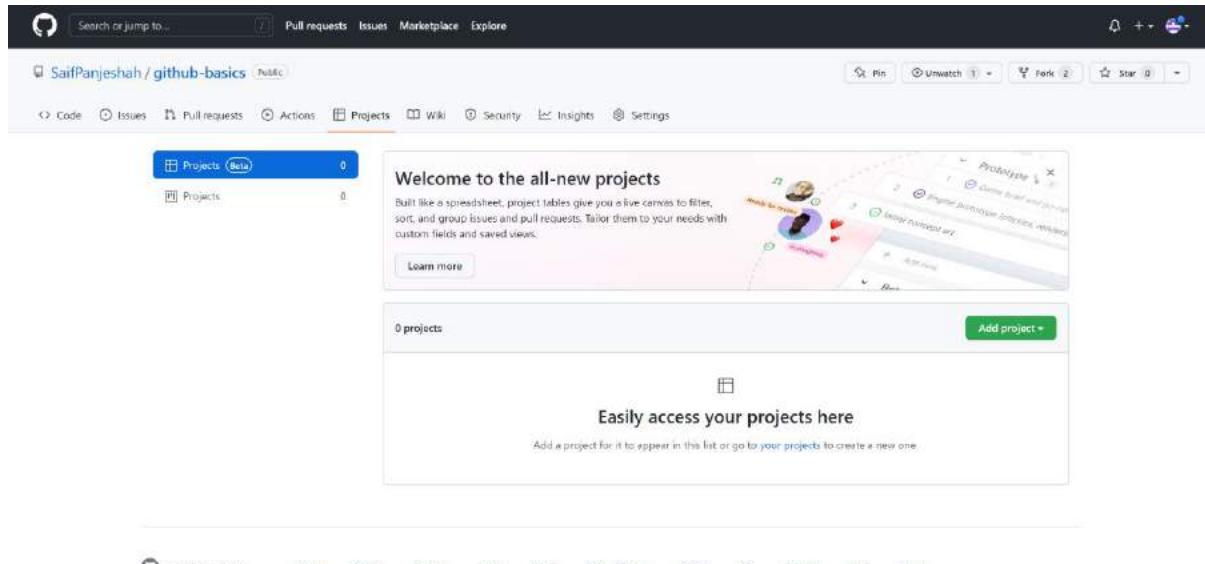


Fig. Project in GitHub

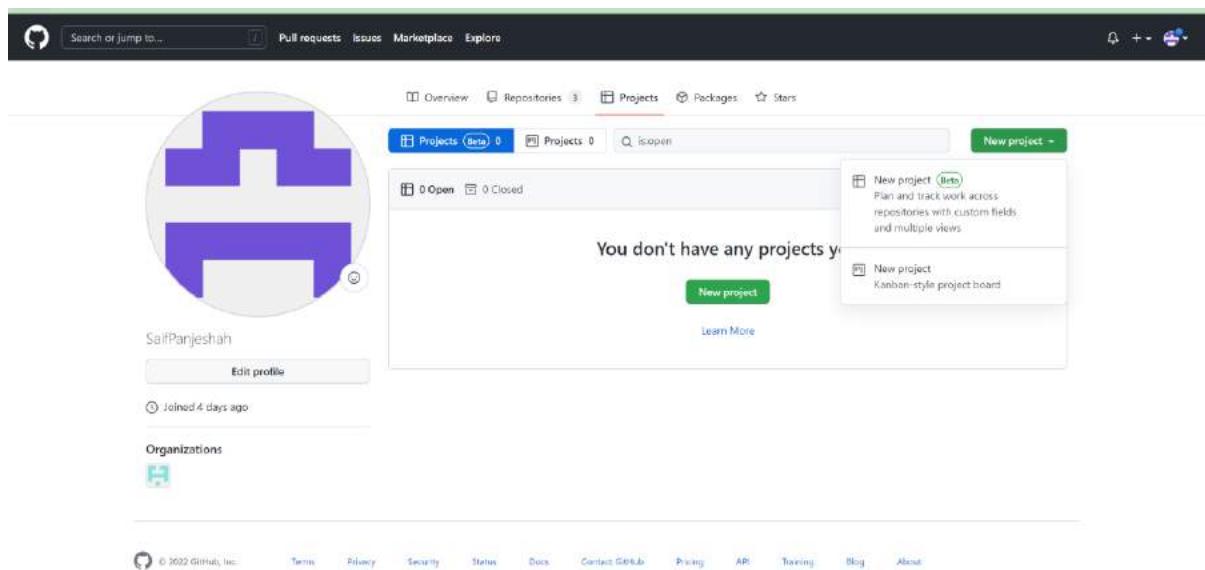


Fig. Creating a Project with Kanban Style

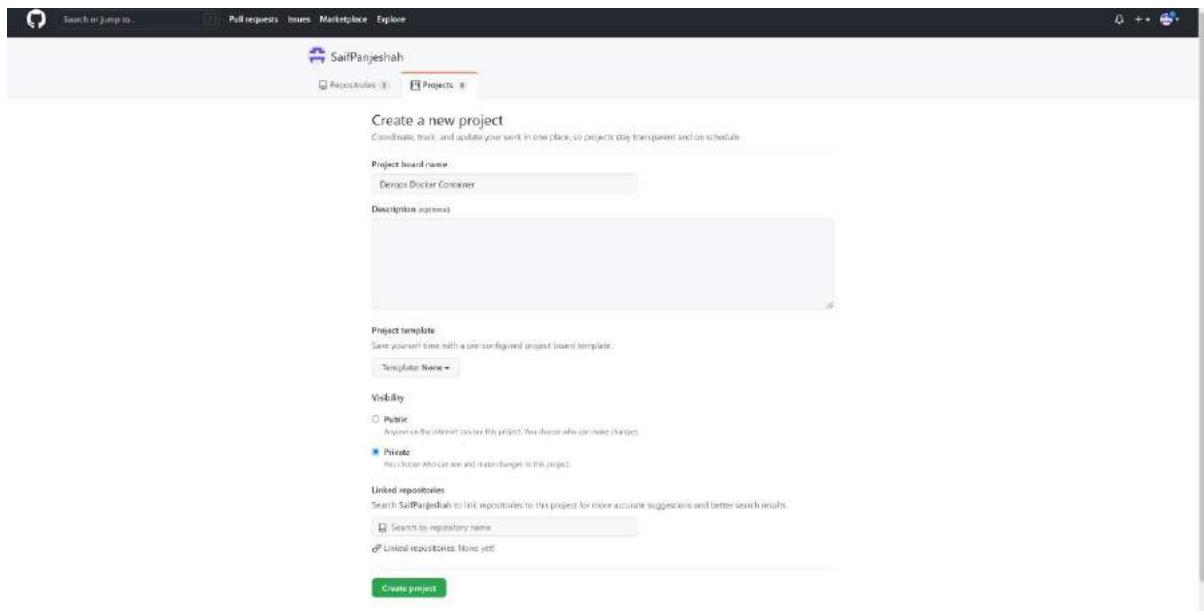


Fig. Devops Docker Container Project

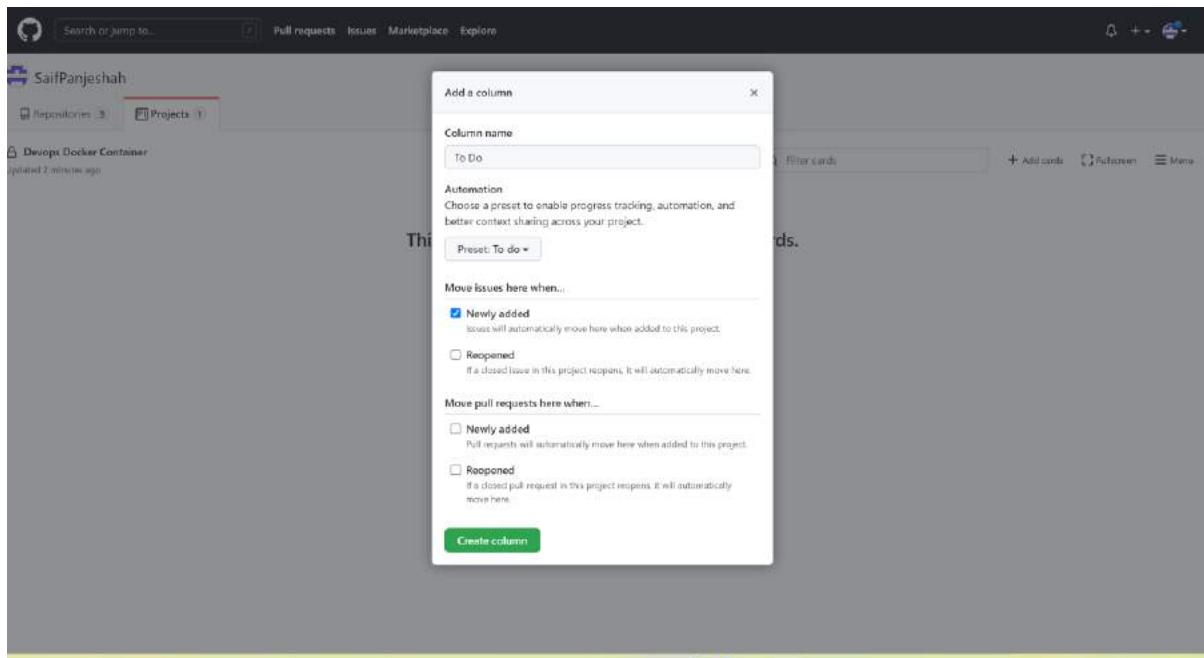


Fig. adding a column what to do

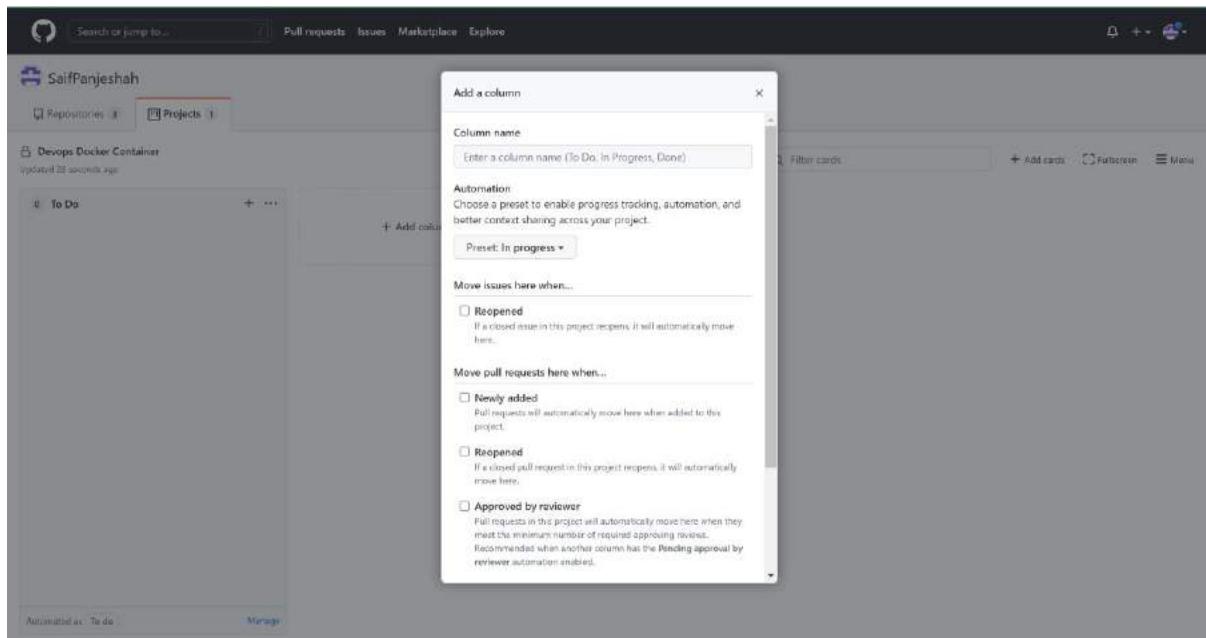


Fig. Creating another column with InProgress

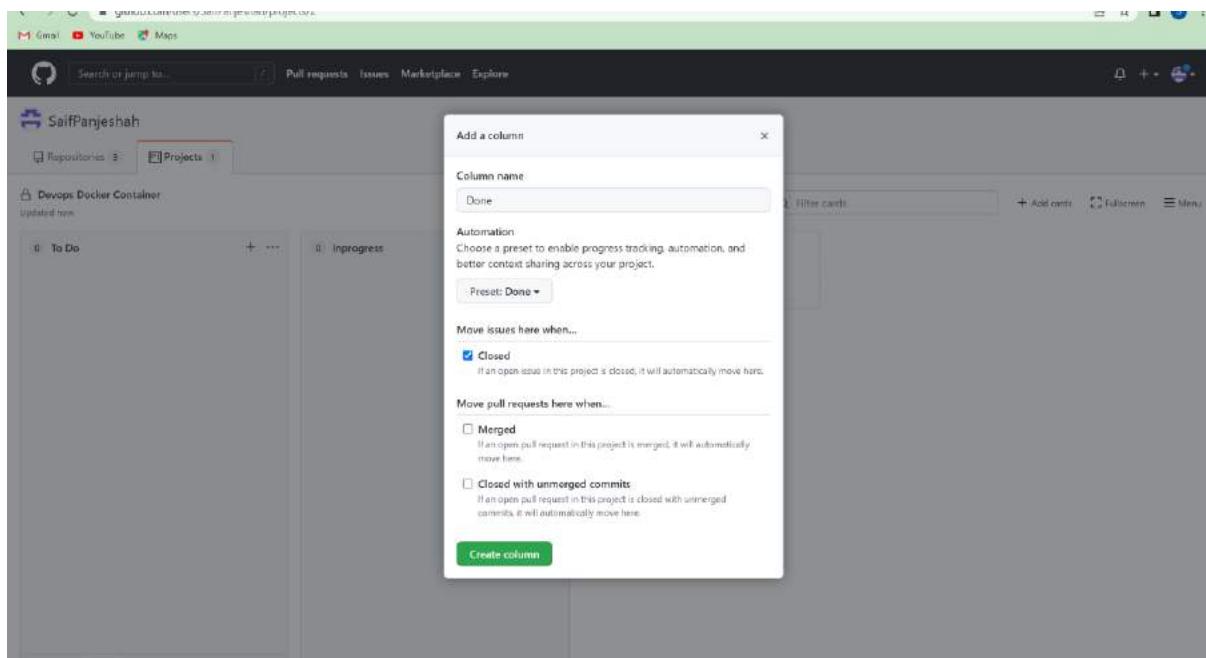


Fig. Final Column with work Done

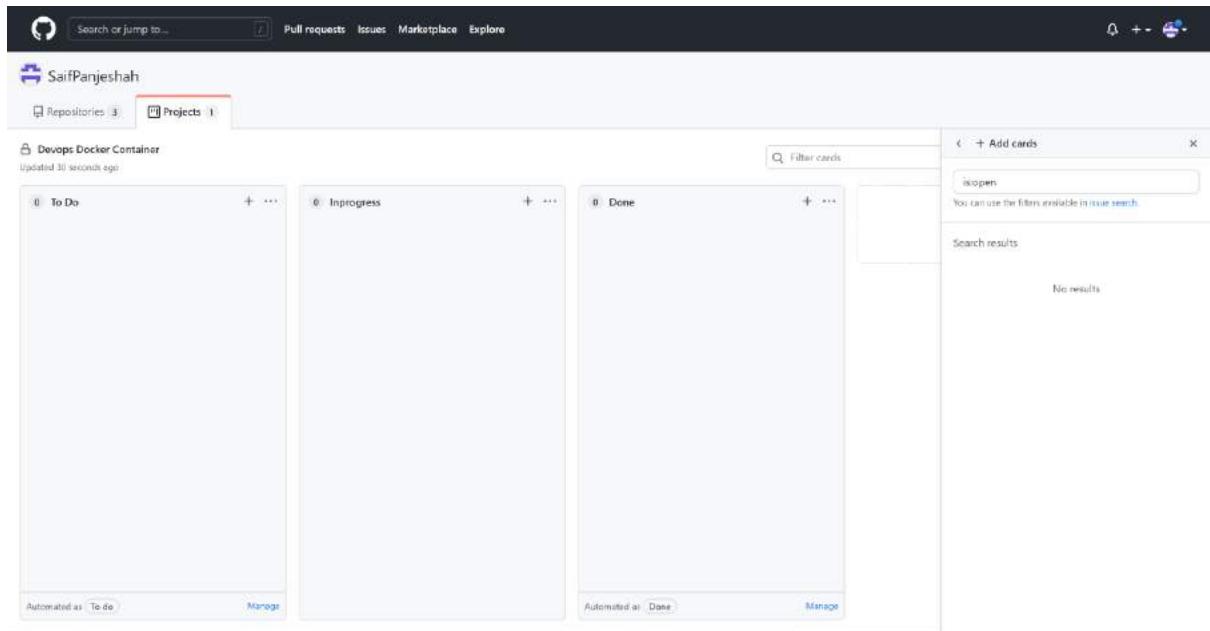


Fig. Project Created successful

Let's assign a work with open Issues on this Project:

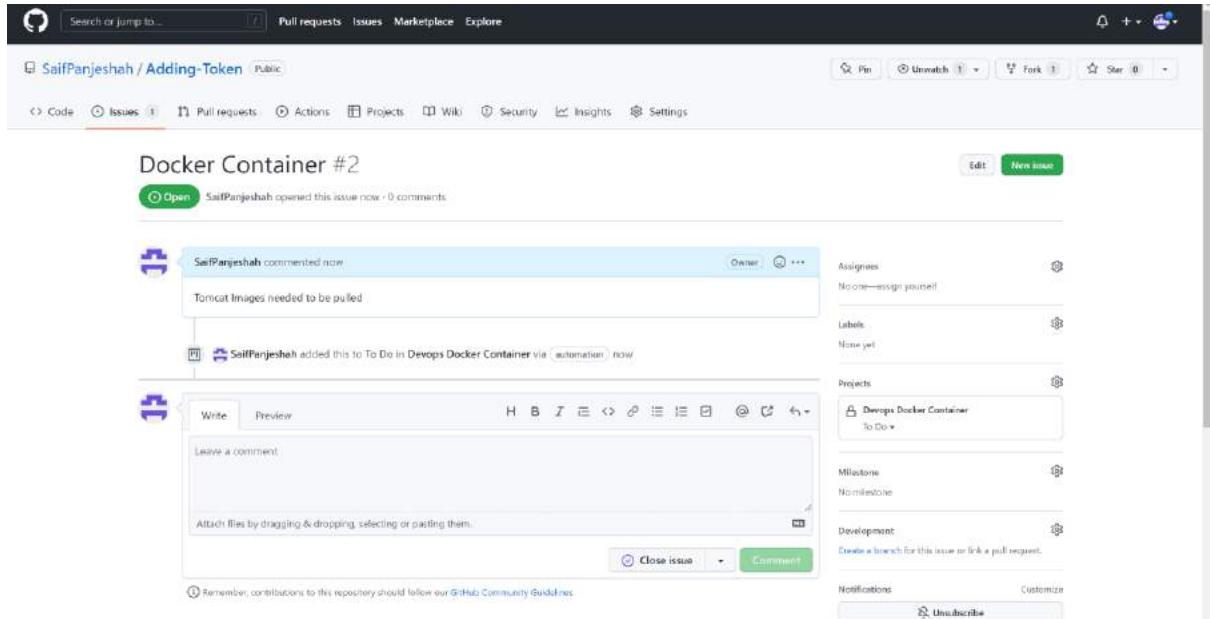


Fig. Creating new Issues

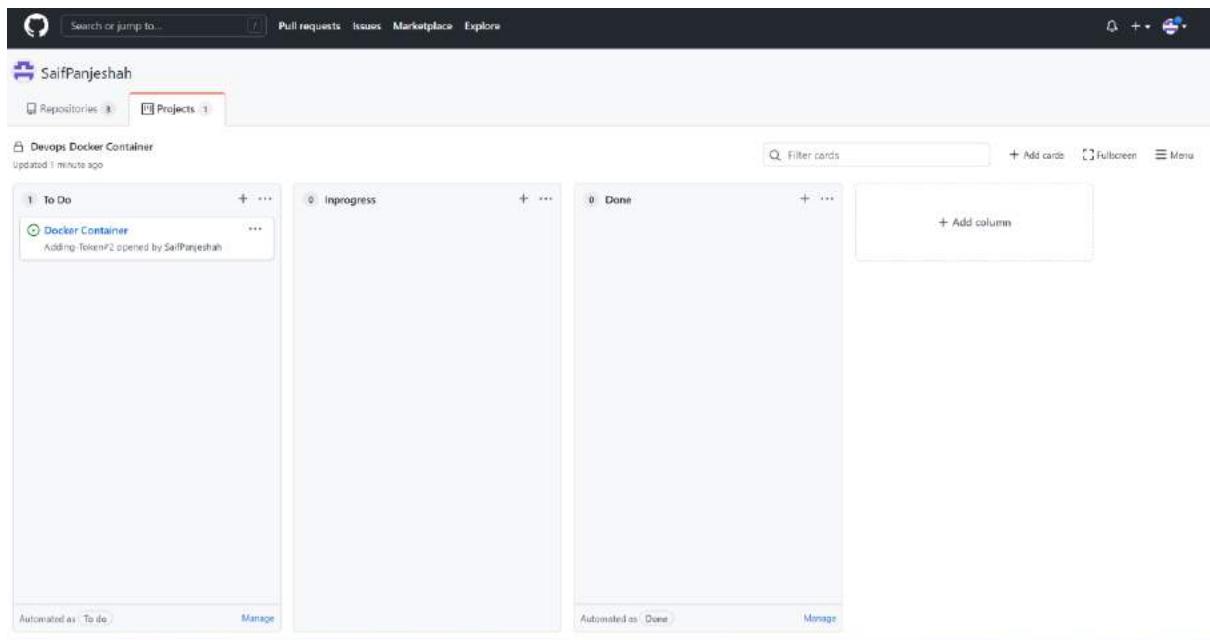


Fig. Card has been successful added to projects

**Close this Issue and create a new Issues Check again the Devops Docker Container Project**

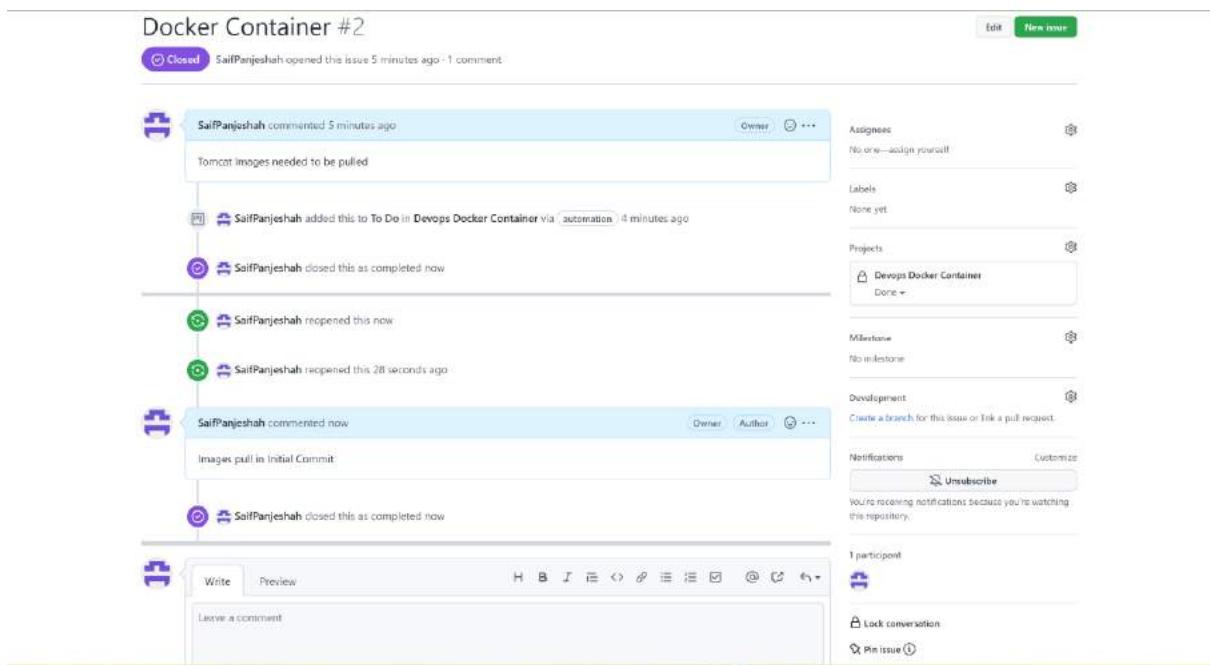


Fig. Closing Issue

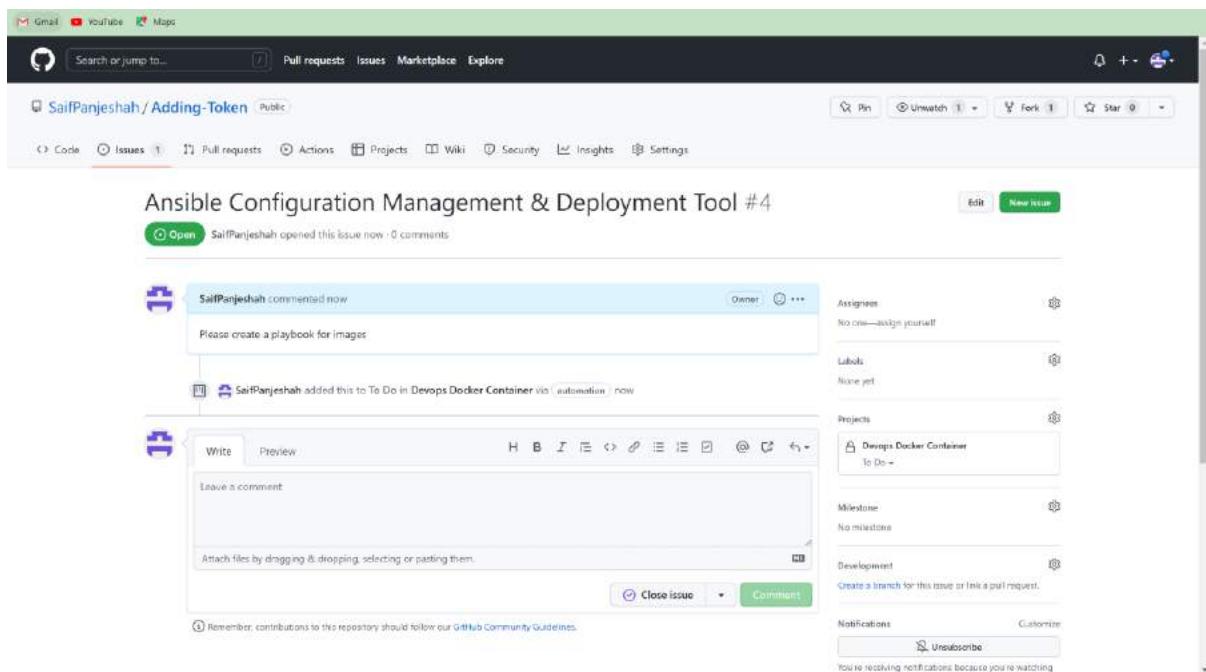


Fig. Creating New Issue

## Let's Explore Devops Docker Container Project:

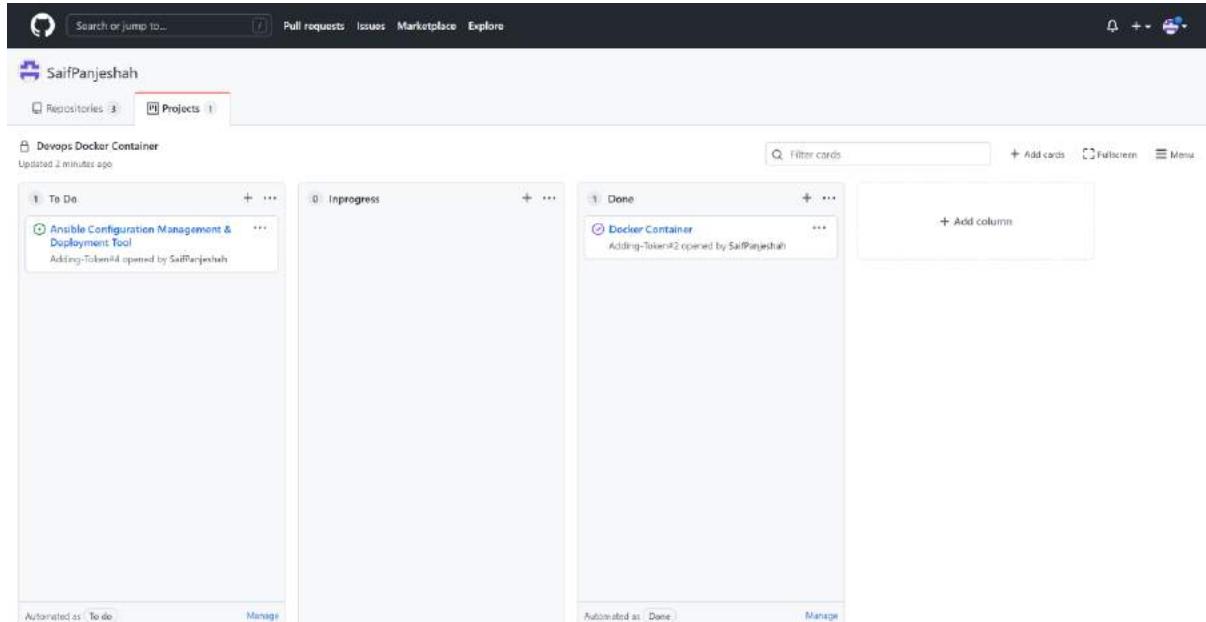
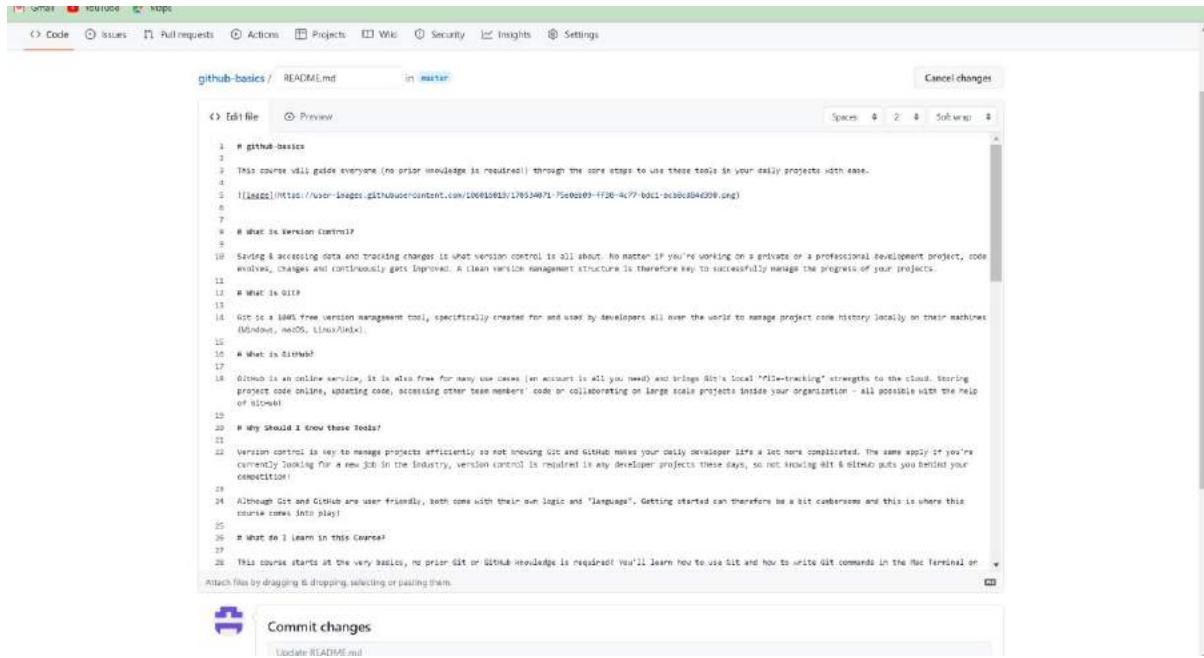


Fig. cards successful created on Devops Docker Container Project

## Creating a README File:

README (as the name suggests: "read me") is the first file one should read when starting a new project. It's a set of useful information about a project, and a kind of manual. A README text file appears in many various places and refers not only to programming.



The screenshot shows the GitHub interface for a repository named 'github-basics'. The user is editing the 'README.md' file in the 'master' branch. The code editor contains the following content:

```
1 # github-basics
2
3 This course will guide everyone (no prior knowledge is required) through the core steps to use these tools in your daily projects with ease.
4
5 
6
7
8 # What is Version Control?
9
10 Saving & accessing data and tracking changes is what version control is all about. No matter if you're working on a private or a professional development project, code evolves, changes and continuously gets improved. A clean version management structure is therefore key to successfully manage the progress of your projects.
11
12 # What is Git?
13
14 Git is a 100% free version management tool, specifically created for and used by developers all over the world to manage project code history locally on their machines (Windows, macOS, Linux/Unix).
15
16 # What is GitHub?
17
18 GitHub is an online service, it is also free for many use cases (an account is all you need) and brings Git's local "file-tracking" strengths to the cloud. Sharing project code online, updating code, accessing other team members' code or collaborating on large scale projects inside your organization - all possible with the help of GitHub.
19
20 # Why Should I Know These Tools?
21
22 Version control is key to manage projects efficiently so not knowing Git and GitHub makes your daily developer life a lot more complicated. The same apply if you're currently looking for a new job in the industry, version control is required in any developer projects these days, so not knowing Git & GitHub puts you behind your competition!
23
24 Although Git and GitHub are user friendly, both come with their own logic and "language". Getting started can therefore be a bit cumbersome and this is where this course comes into play!
25
26 # What do I Learn in This Course?
27
28 This course starts at the very basics, no prior Git or GitHub knowledge is required! You'll learn how to use Git and how to write Git commands in the Mac Terminal or attach files by dragging & dropping, selecting or pasting them.
```

At the bottom of the editor, there is a 'Commit changes' button with a blue icon and a 'Update README.md' link.

Fig. Creating a README File

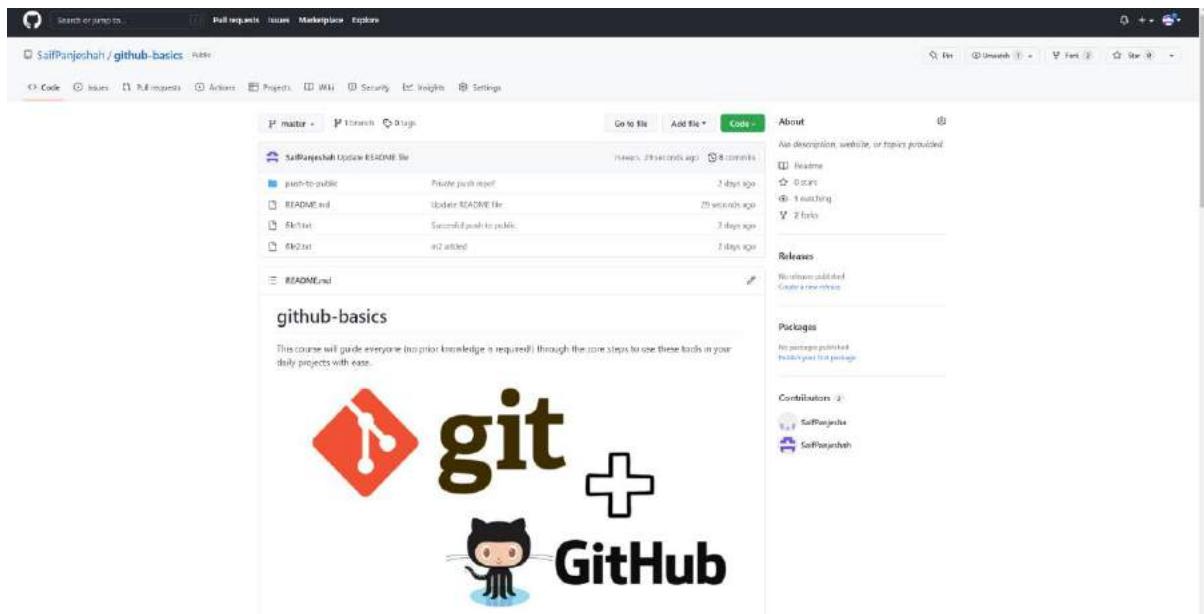


Fig. Complete README File added

## Presenting Yourself as a Developer on GitHub:

### Making GitHub Page more attractive.

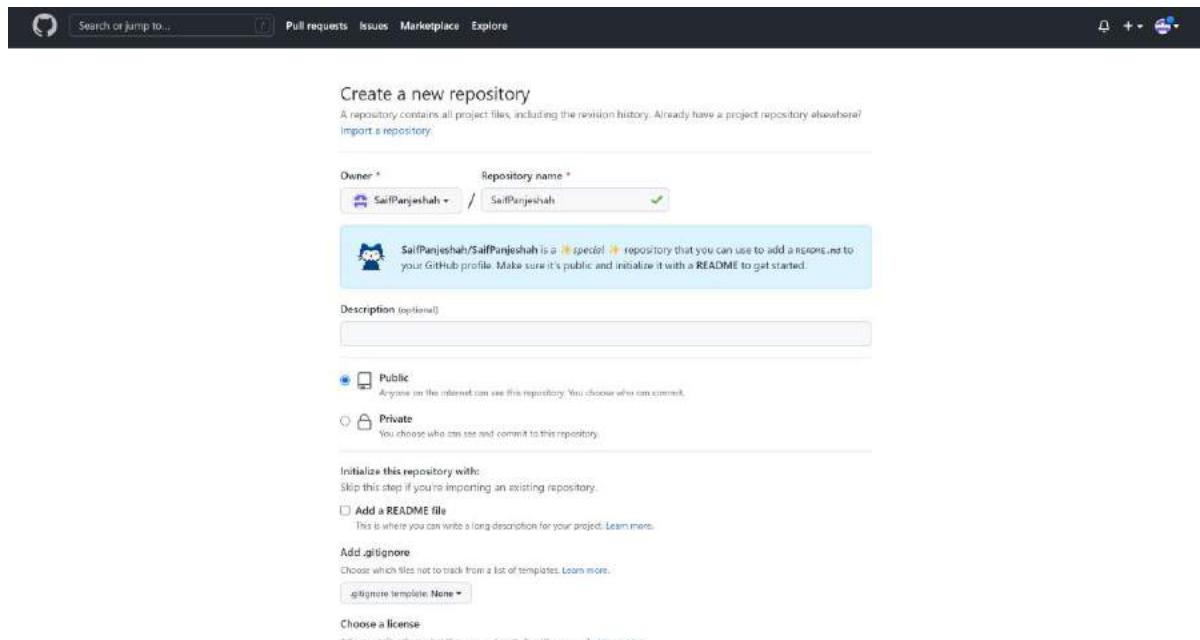


Fig. Creating Special secret repository with owner name

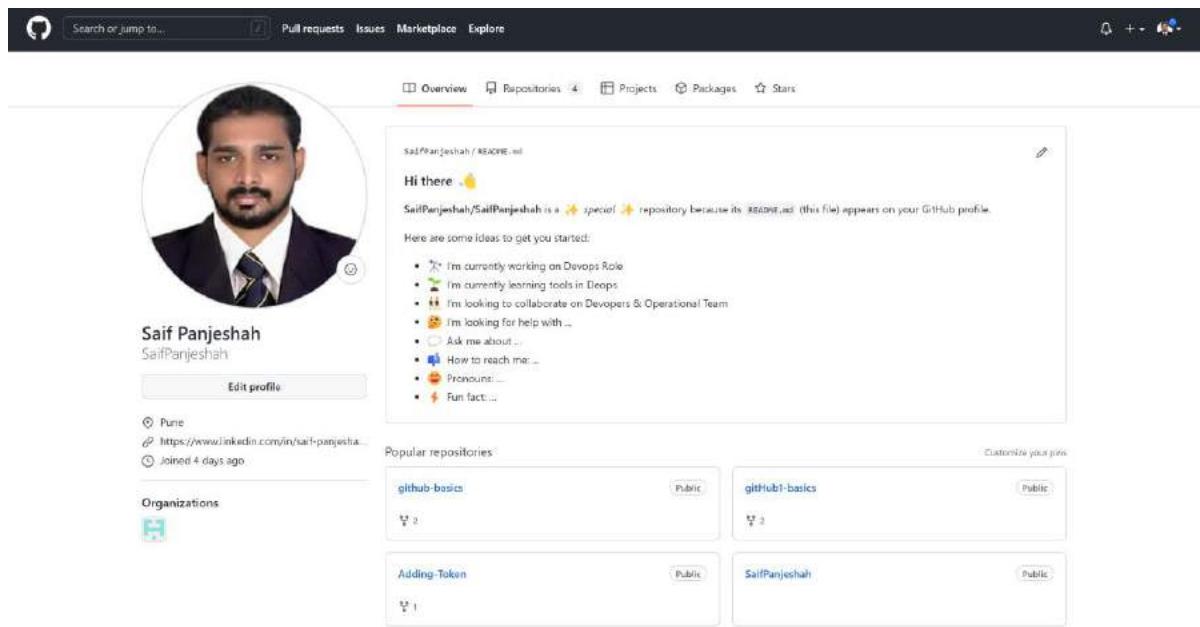


Fig. Presenting Yourself as a Developer on GitHub

## About GitHub Stars:

Stars are all about likes on social media “Git and GitHub Basics”.

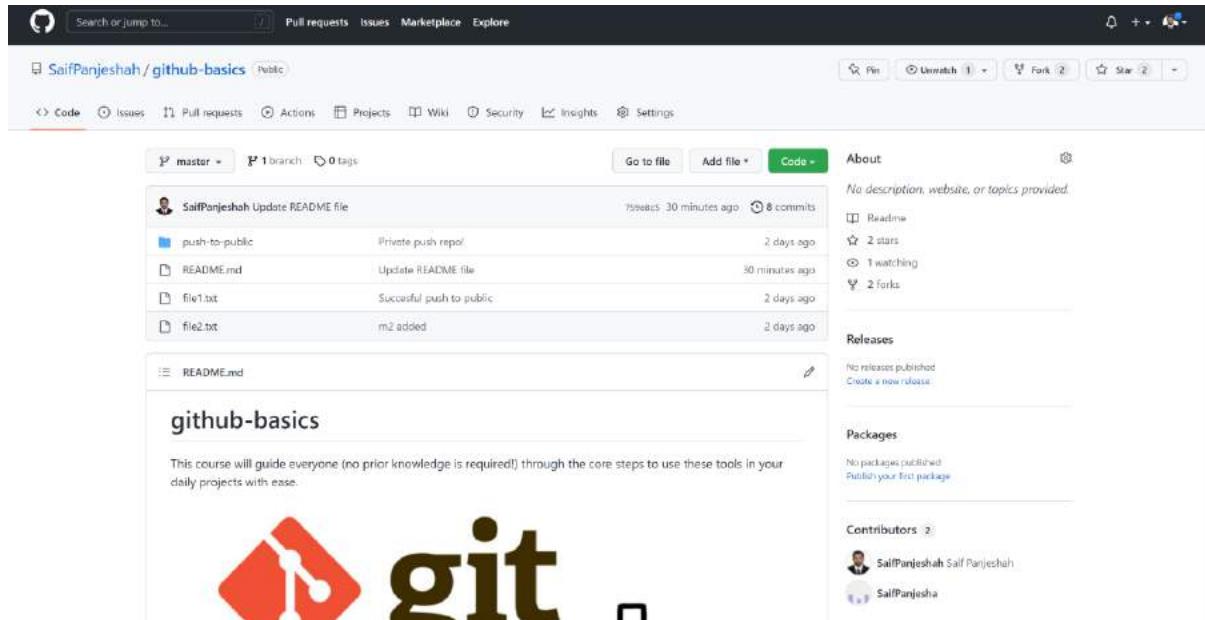


Fig. GitHub Stars

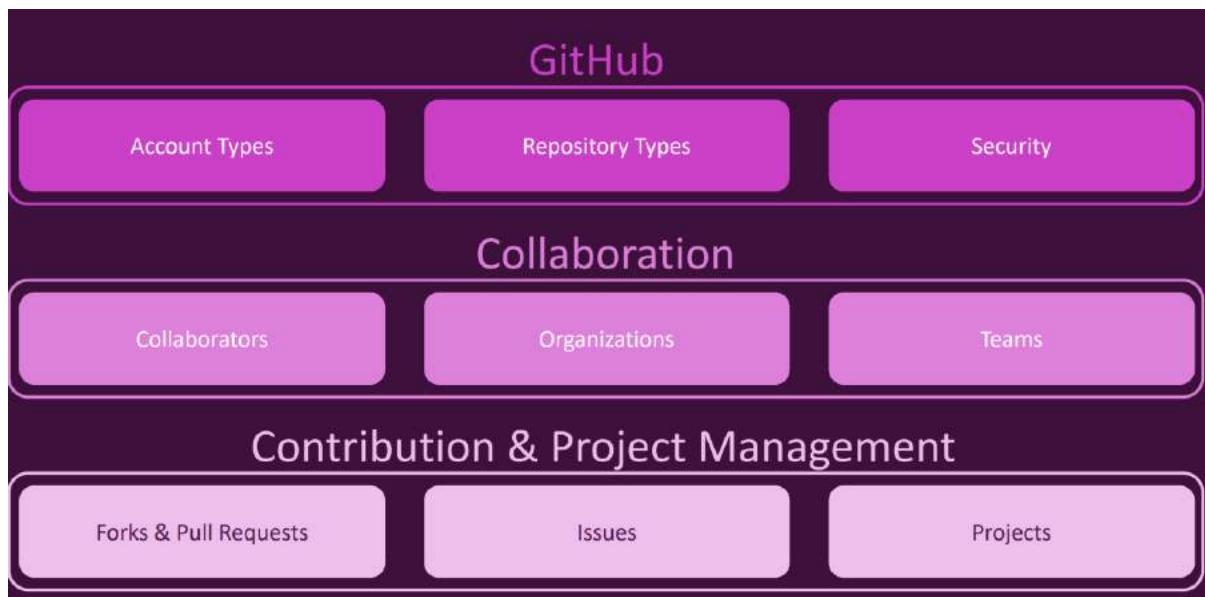


Fig. Wrap up Model Summary

## **Useful Resources and Link:**

More about Permission Levels for User Account Repositories

=> <https://docs.github.com/en/github/setting-up-and-managing-your-github-user-account/managing-user-account-settings/permission-levels-for-a-user-account-repository>

VueJS GitHub Page => <https://github.com/vuejs/vue>