

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.

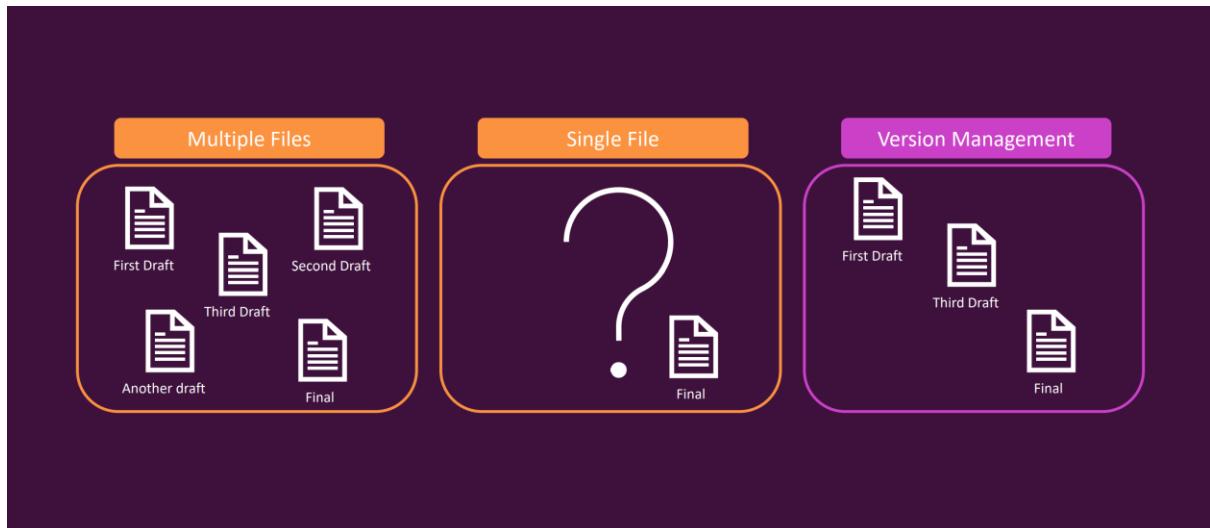


Fig. Version Management / Control



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?)

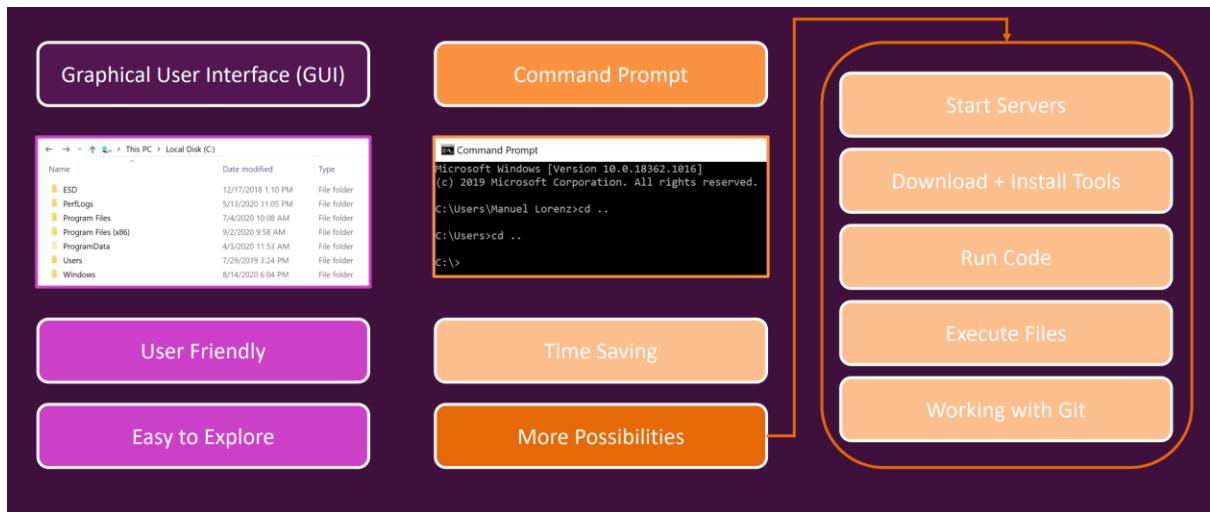


Fig. GUI vs Command Prompt

Comparing Mac & Windows Command Line:

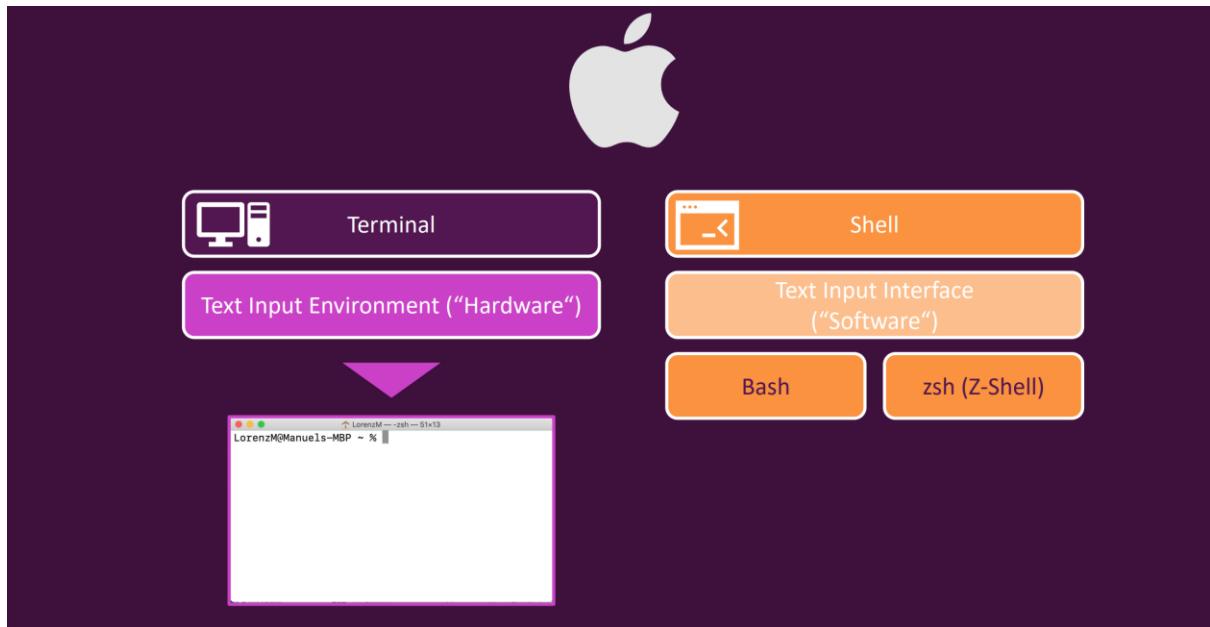


Fig. Mac Terminology

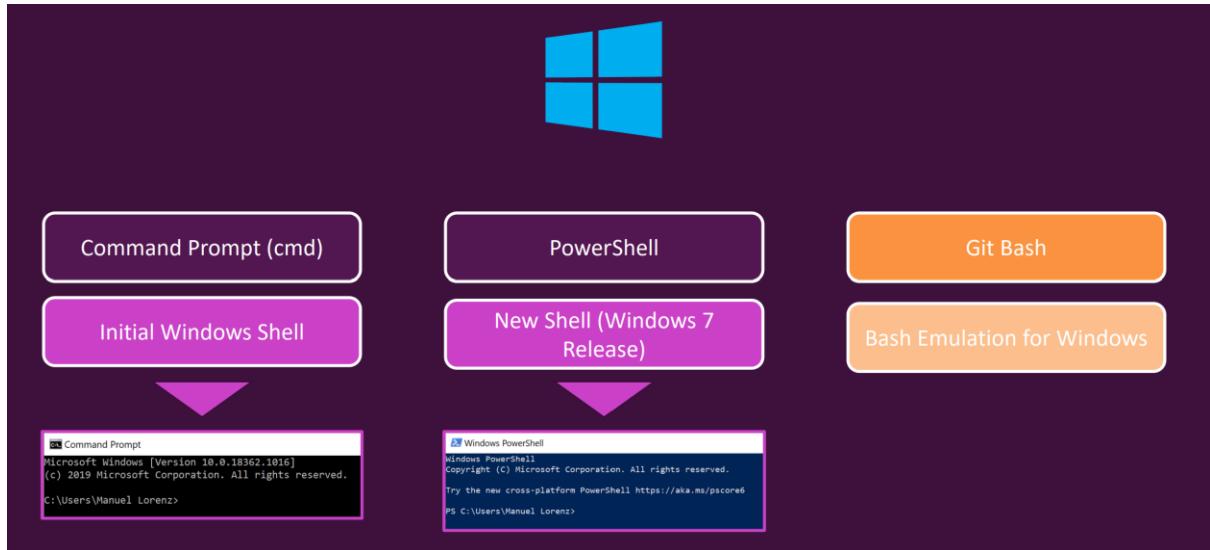


Fig. Windows Terminology

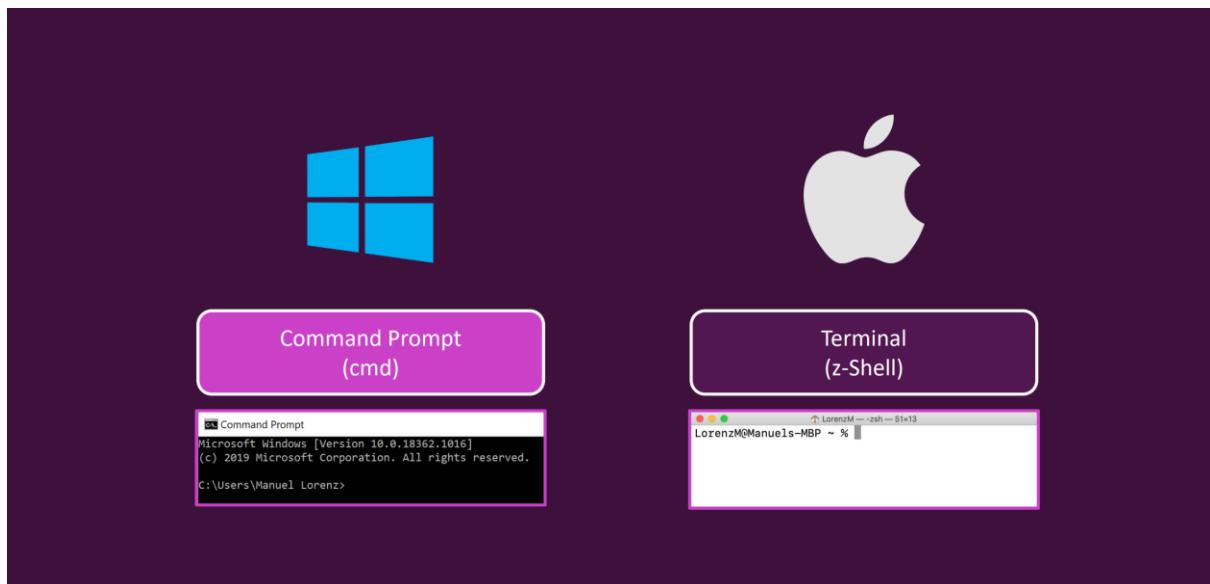


Fig. Command Line Tools

Mac Terminal:

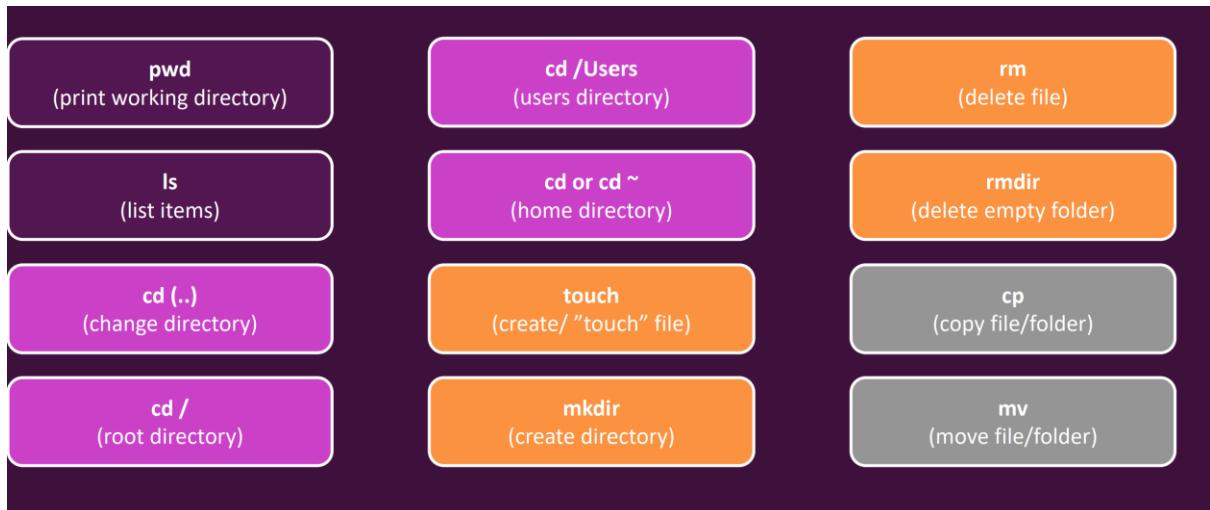


Fig. Mac Terminal (z-Shell Command)

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

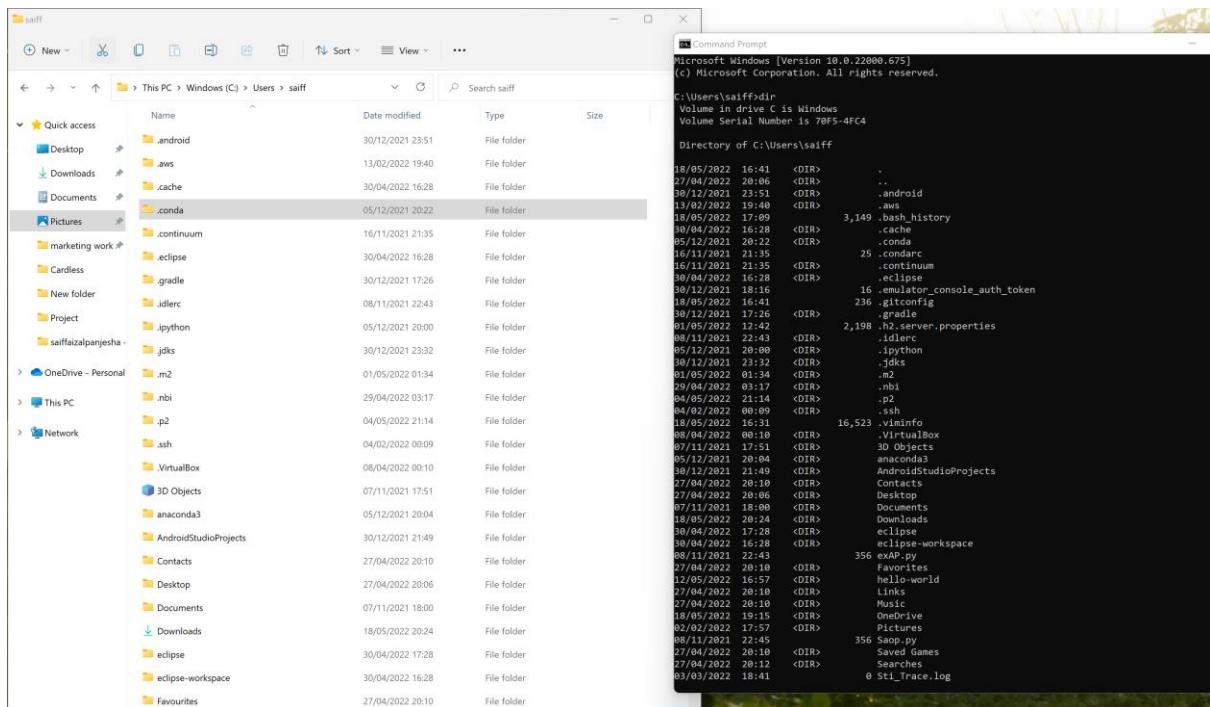


Fig. Lists of Items

- `cd(..)`: Change Directory

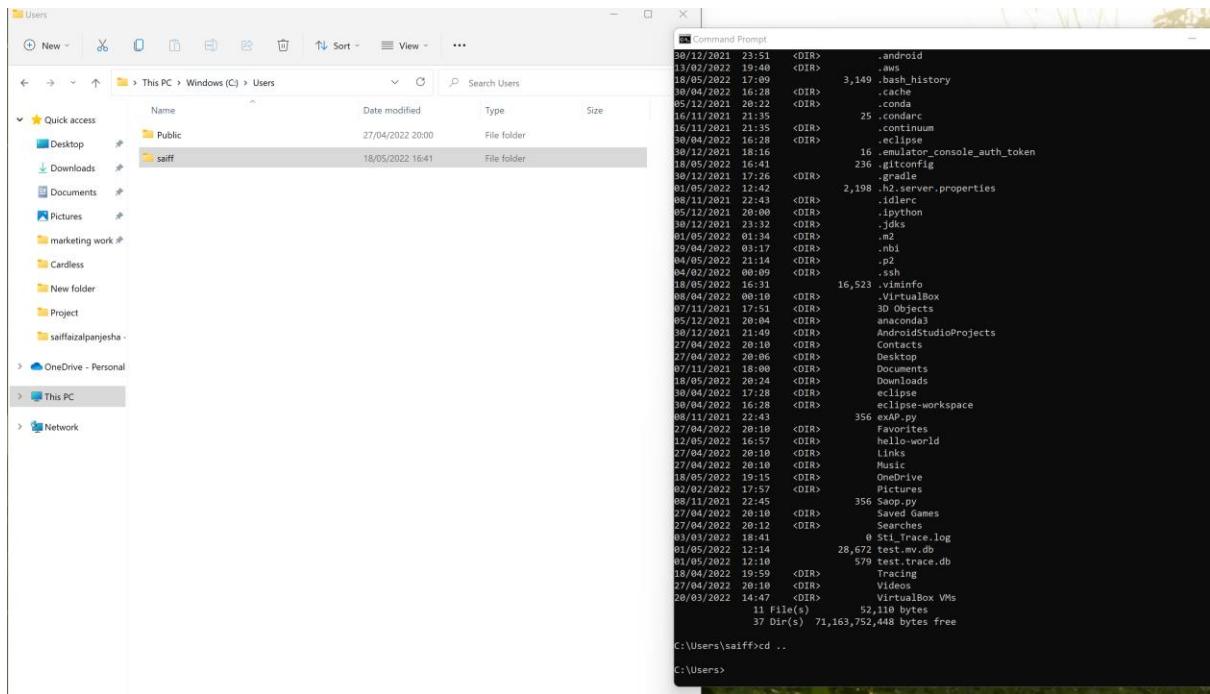


Fig. Change Directory to Users

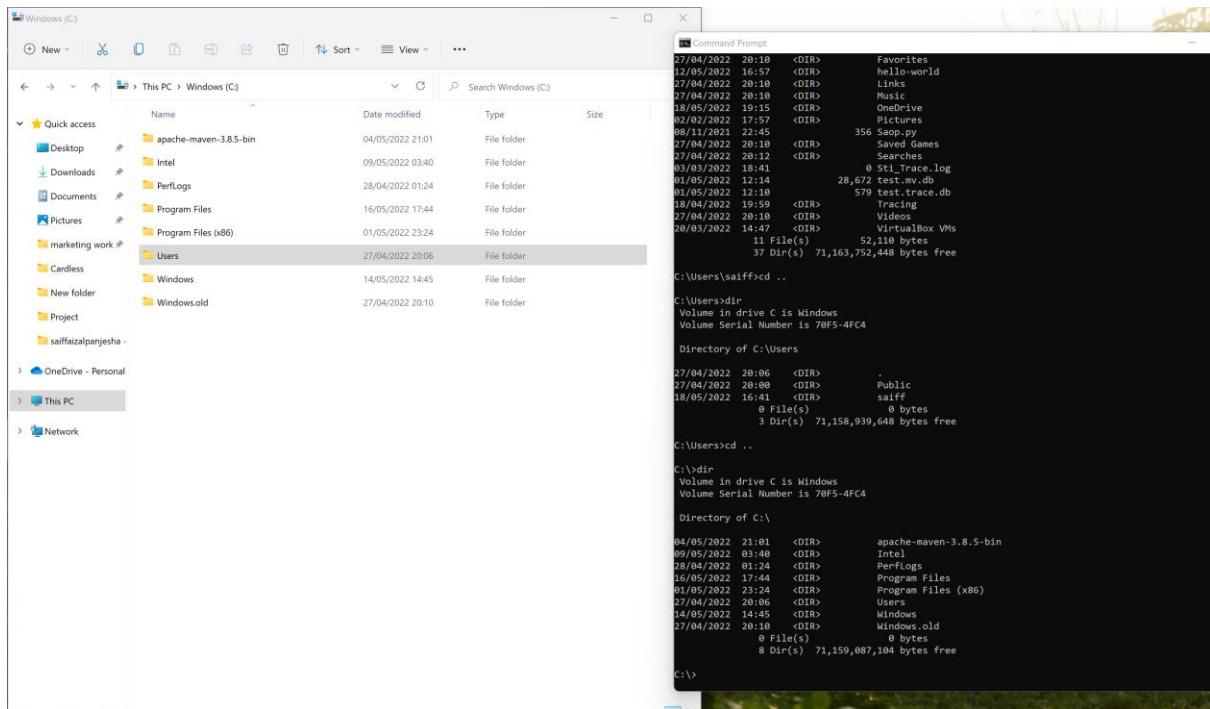


Fig. Change Directory to C drive

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
```

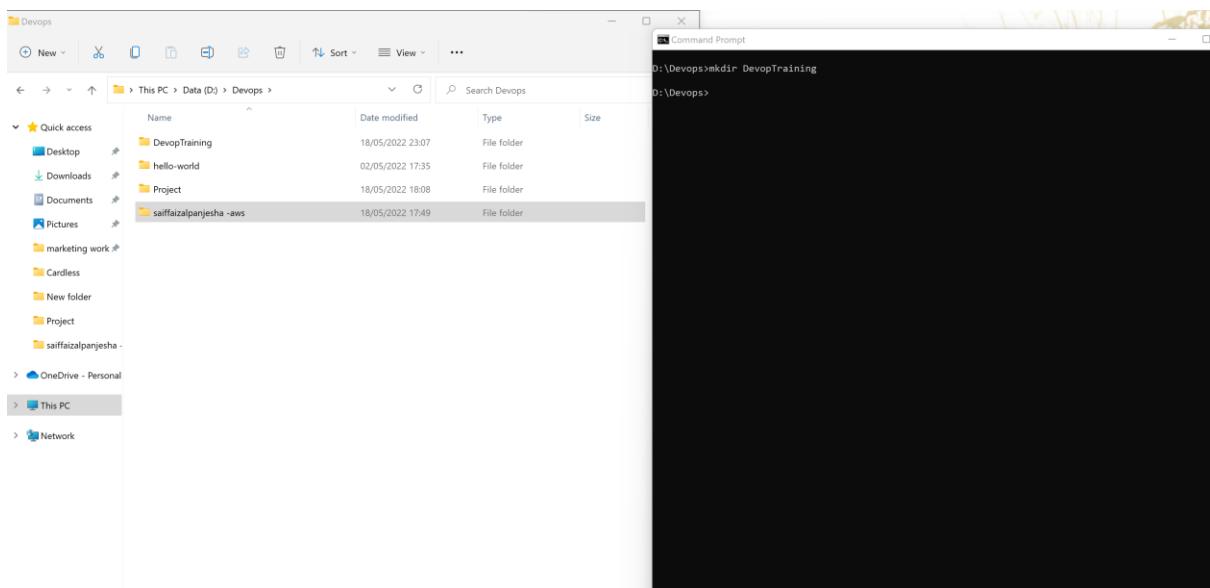
```
D:\Devops\saiffaizalpanjesha -aws>dir  
Volume in drive D is Data  
Volume Serial Number is 04F5-839B  
Directory of D:\Devops\saiffaizalpanjesha -aws  
18/05/2022 17:49 <DIR> .  
18/05/2022 18:08 <DIR> ..  
12/04/2022 08:49 26,740 3646_Skills.pdf  
18/05/2022 17:49 252 Devops Other Courses.txt  
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 04F5-839B

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>        saiffaizalpanjesta -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 04F5-839B

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 04F5-839B

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>        saiffaizalpanjesta -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 04F5-839B

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?

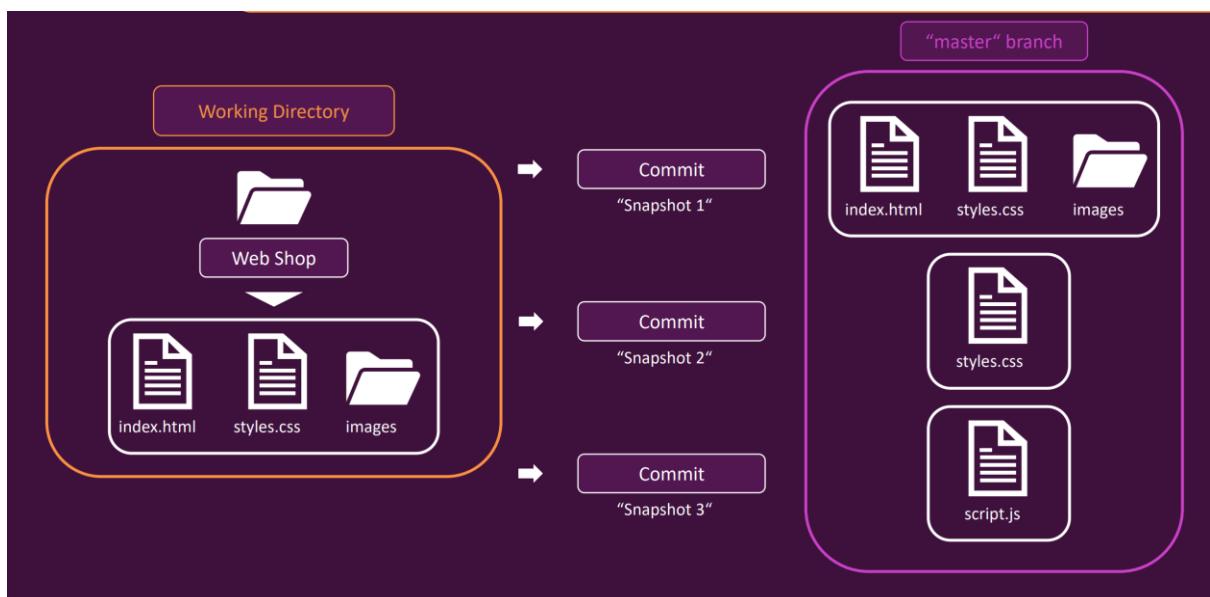


Fig. Working of Git.

Working Directory vs Repository

Git under the Hood:

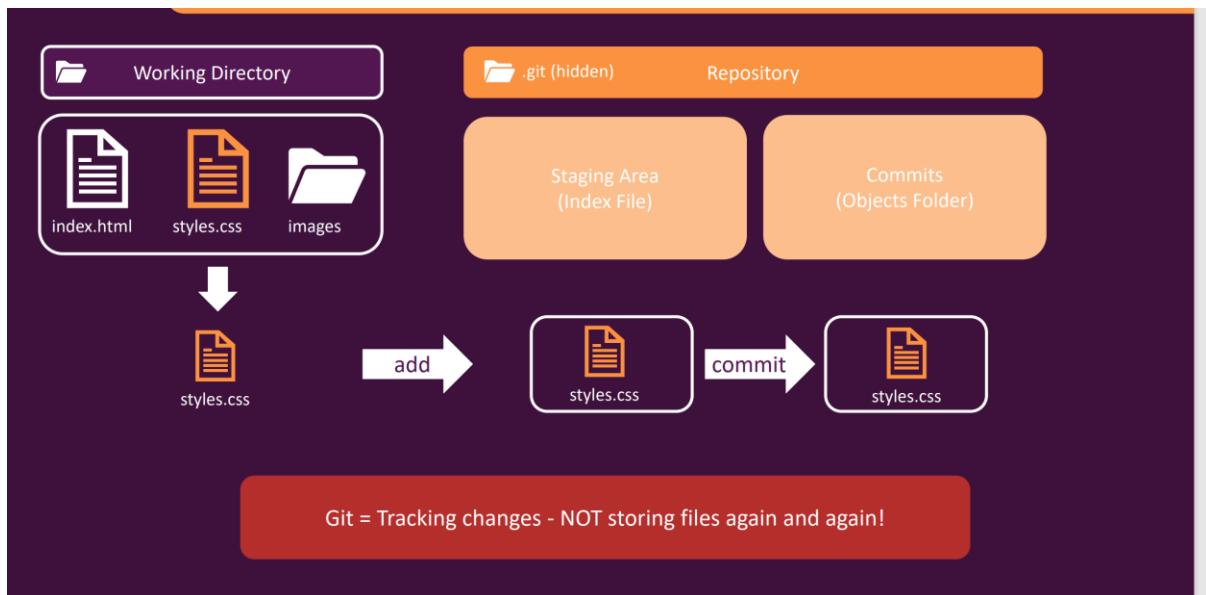


Fig. Git Under the Hood (Directory & Repositories)

Understanding Branches & Commit

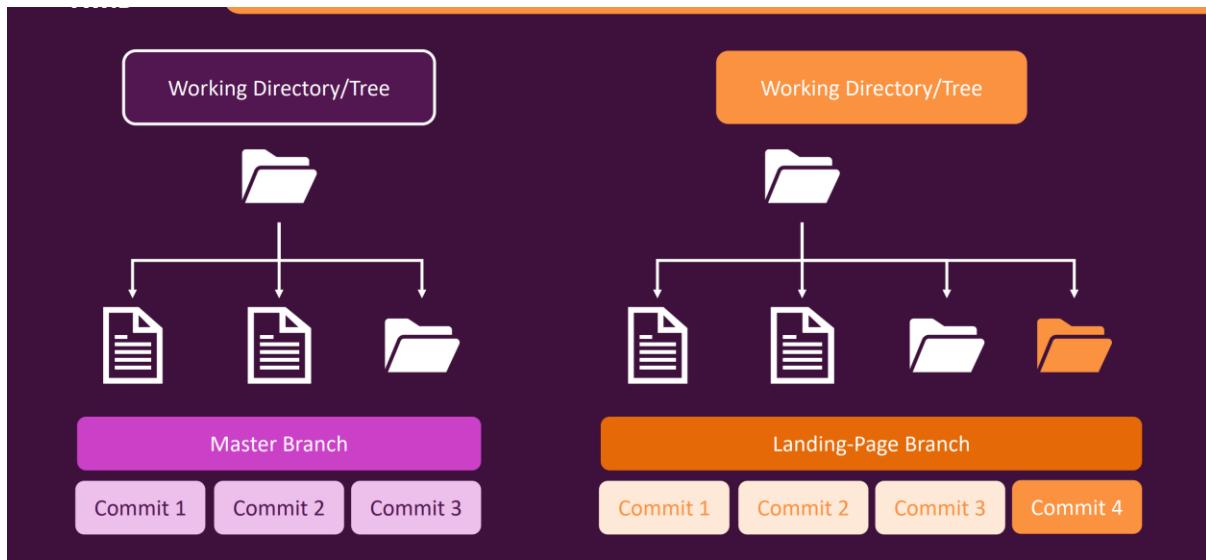
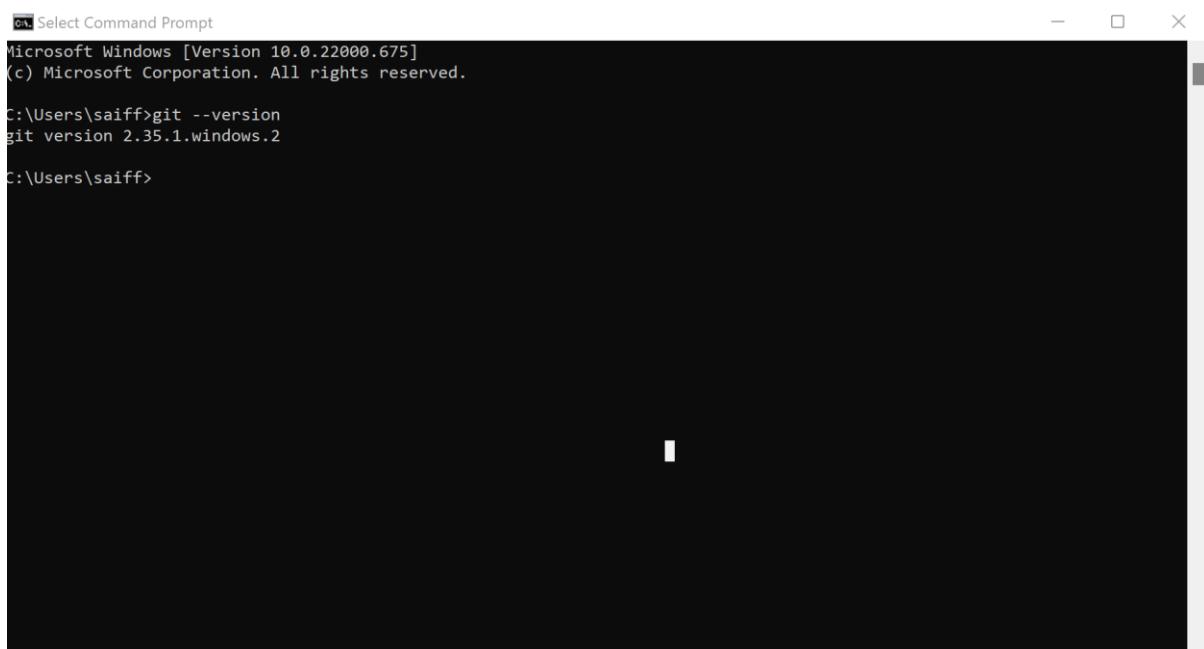


Fig. Branches & Commit

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.



```
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.”

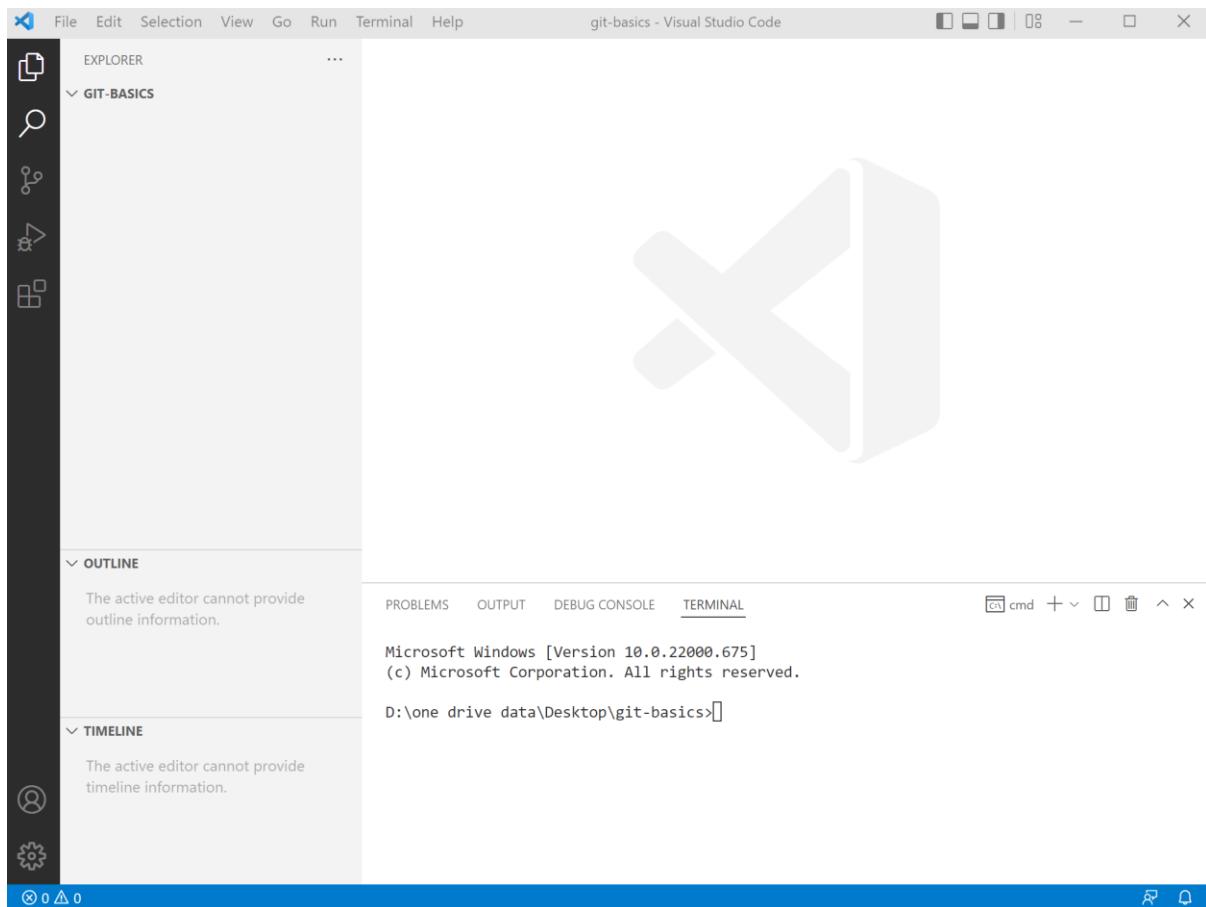


Fig. Visual Studio Install Success

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/
```

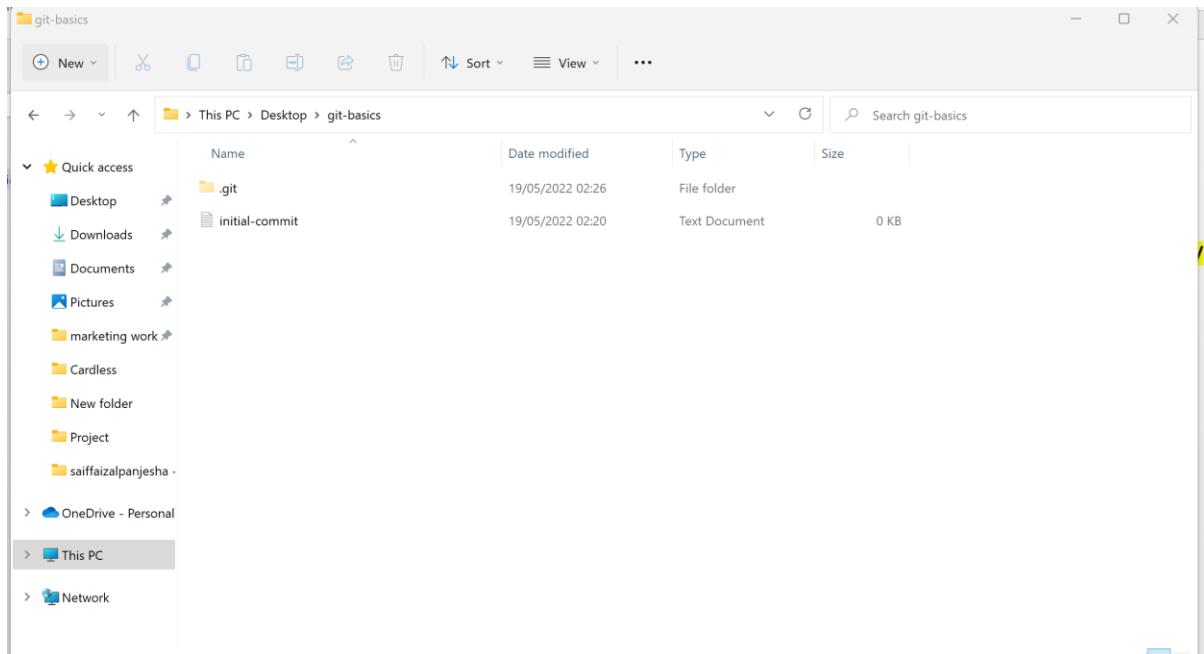


Fig. git hidden folder created after initialized

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:

```
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 two files: 'initial-commit.txt' and 'second-commit.txt'. The 'TERMINAL' tab shows 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 shows 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

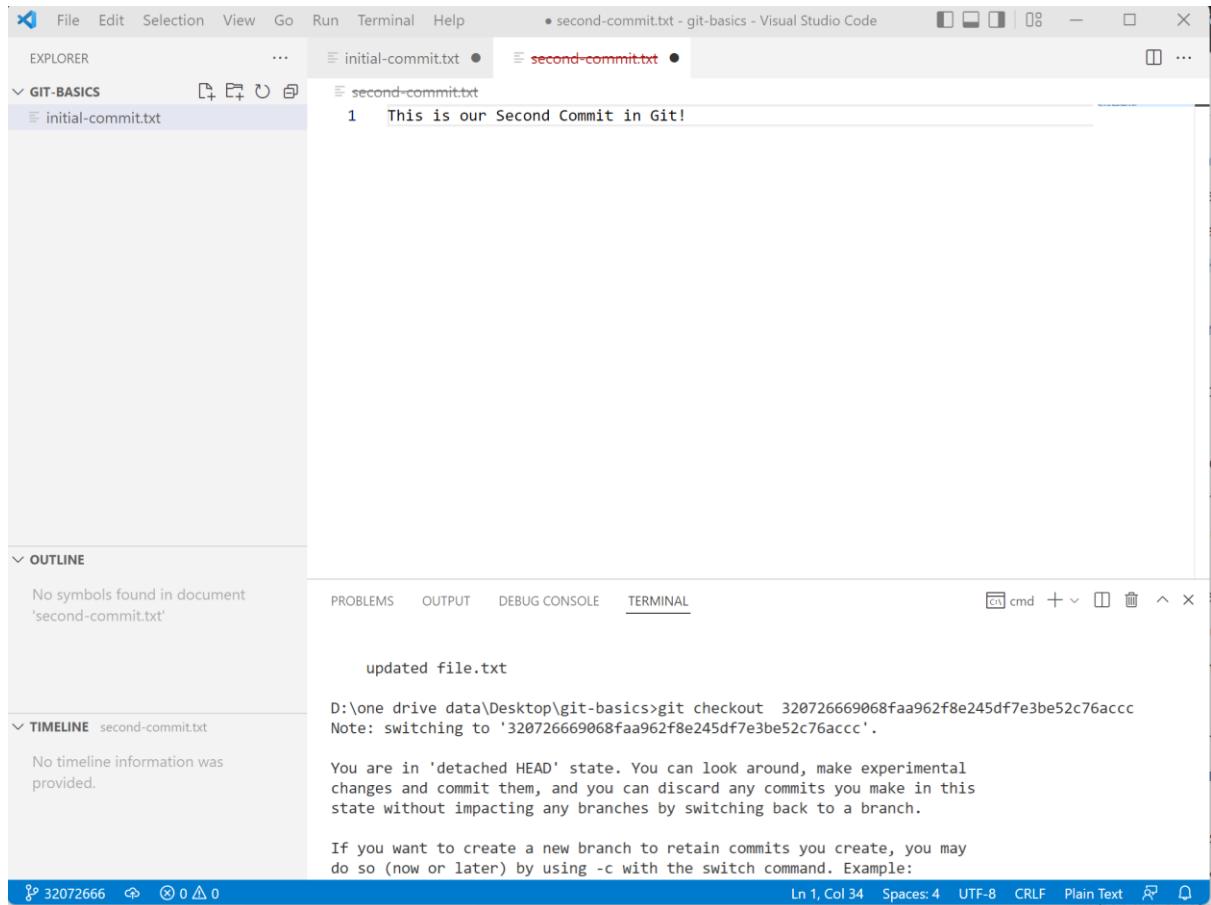


Fig. “Checkout our initial commit”

```
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

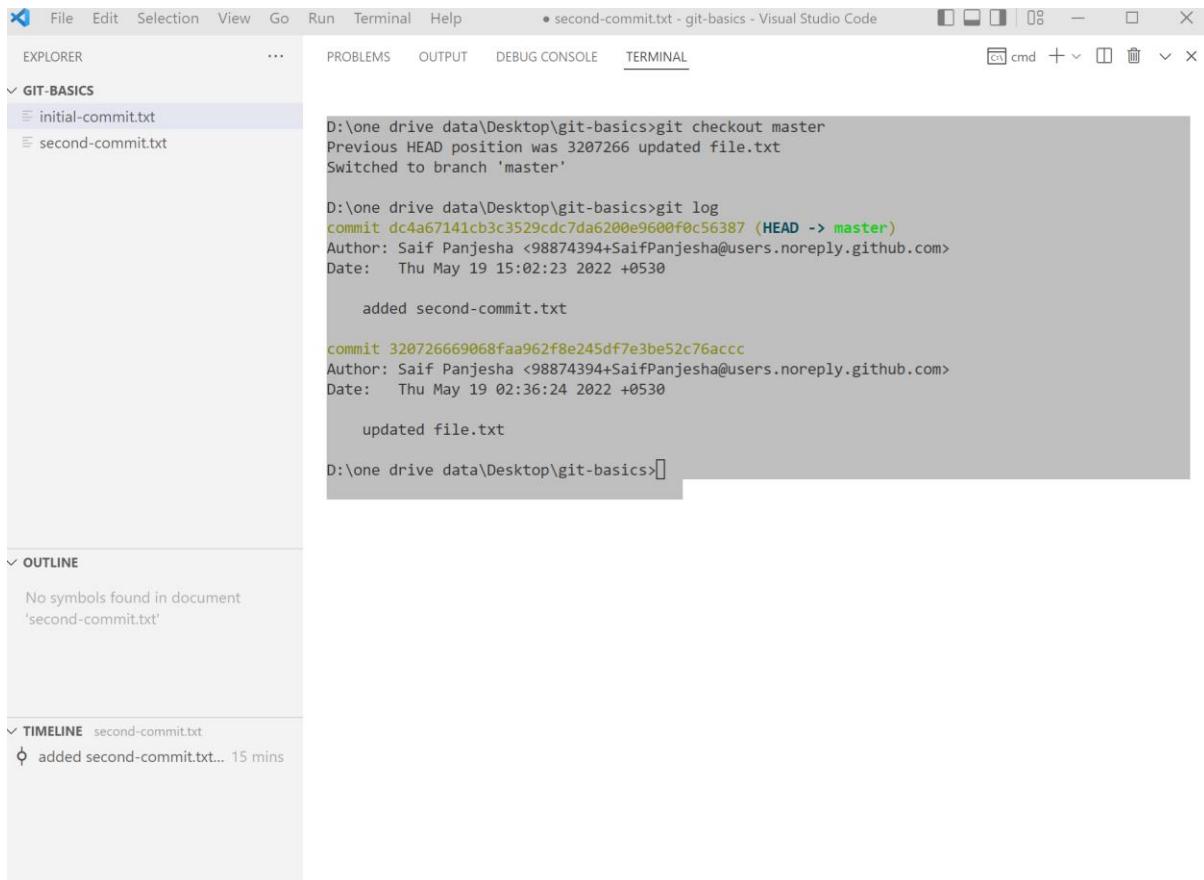
D:\one drive data\Desktop\git-basics>
```

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

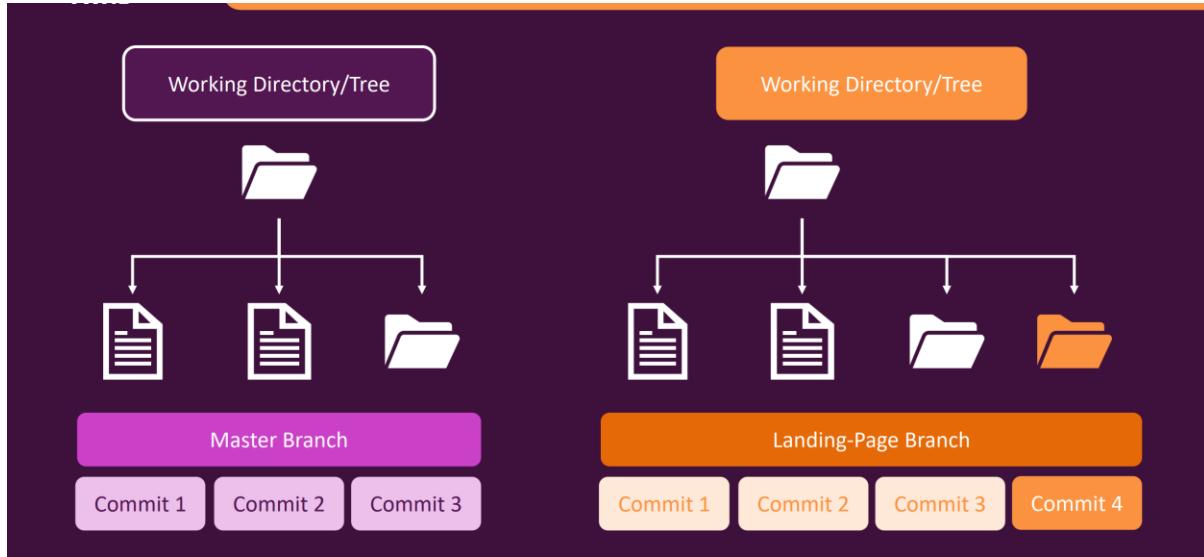


Fig. Goal to create Such copy of master branches

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 Explorer sidebar shows a folder named 'GIT-BASICS' containing two files: 'initial-commit.txt' and 'second-commit.txt'. The Timeline sidebar shows a single commit: 'added second-commit.txt... 15 mins'. The status bar at the bottom indicates the current branch is 'master'.

Fig. Default Branch in current project-* master

Creating Second Branch Landing-Page Branch

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

```
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 with the 'TERMINAL' tab selected. The terminal window displays the following git commands and their output:

```
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 displaying the following git commands and their output:

```
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 has a "GIT-BASICS" section with files "initial-commit.txt", "second-commit.txt", and "Working_with_Branches.txt". The main area is a terminal window with the following output:

```
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

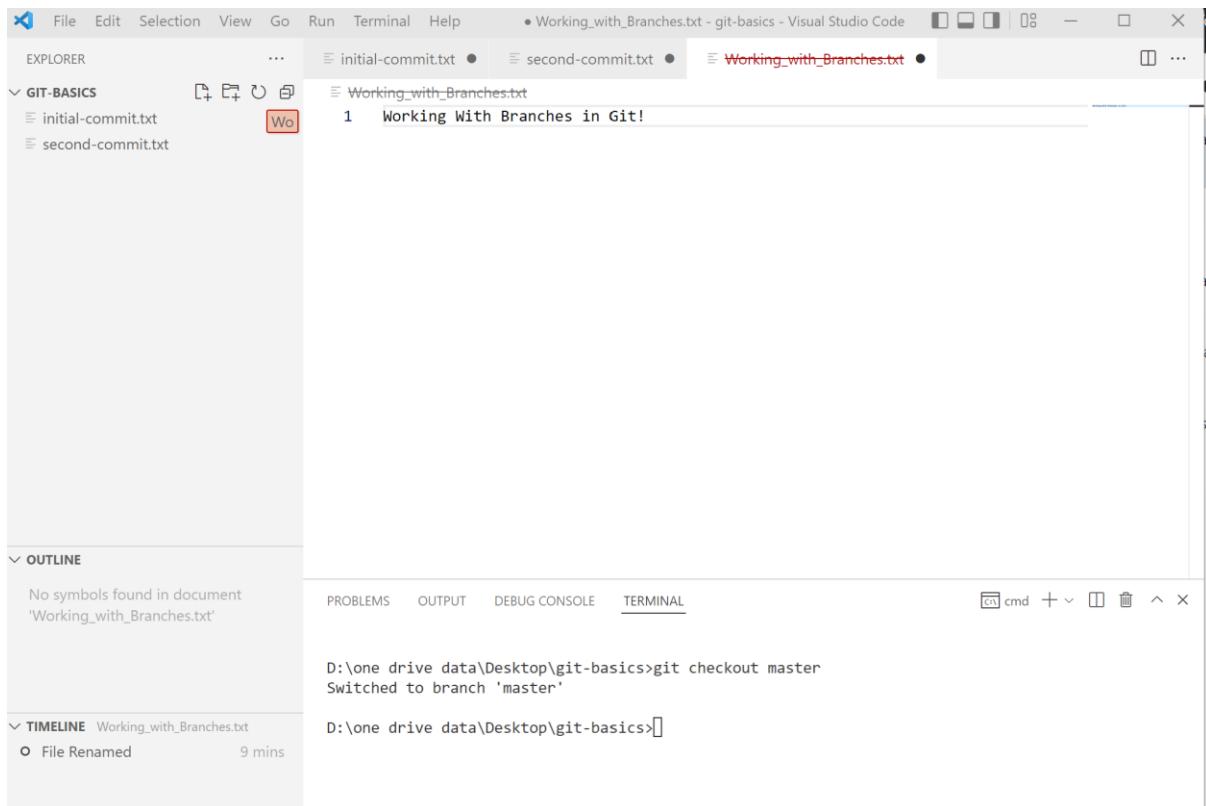


Fig. Switch to Master Branch

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.
- Title Bar:** Working_with_Branches.txt - git-basics - Visual Studio Code.
- Explorer:** Shows three files: initial-commit.txt, second-commit.txt, and Working_with_Branches.txt (which is currently selected).
- Terminal:** Shows the command history:

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

D:\one drive data\Desktop\git-basics>git merge quickfix
Updating dc4a671..cd8816c
Fast-forward
  Working_with_Branches.txt | 0
  1 file changed, 0 insertions(+), 0 deletions(-)
  create mode 100644 Working_with_Branches.txt
```
- Bottom Status Bar:** master, Line 1, Col 30, Spaces: 4, UTF-8, CRLF, Plain Text.

Fig. Merging with Branch

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

- File Explorer:** Shows files: initial-commit.txt, second-commit.txt, and Working_with_Branches.txt.
- Terminal:** The tab is selected, showing the command `git log` and its output:

```
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:** Shows "No symbols found in document 'Working_with_Branches.txt'".
- Timeline:** Shows activity for "Working_with_Branches.txt":
 - Branches are Working (Sai... 10 mins ago)
 - File Renamed (13 mins ago)

Fig. Success History of Merging.

Understanding the HEAD

The Latest commit is known as HEAD.

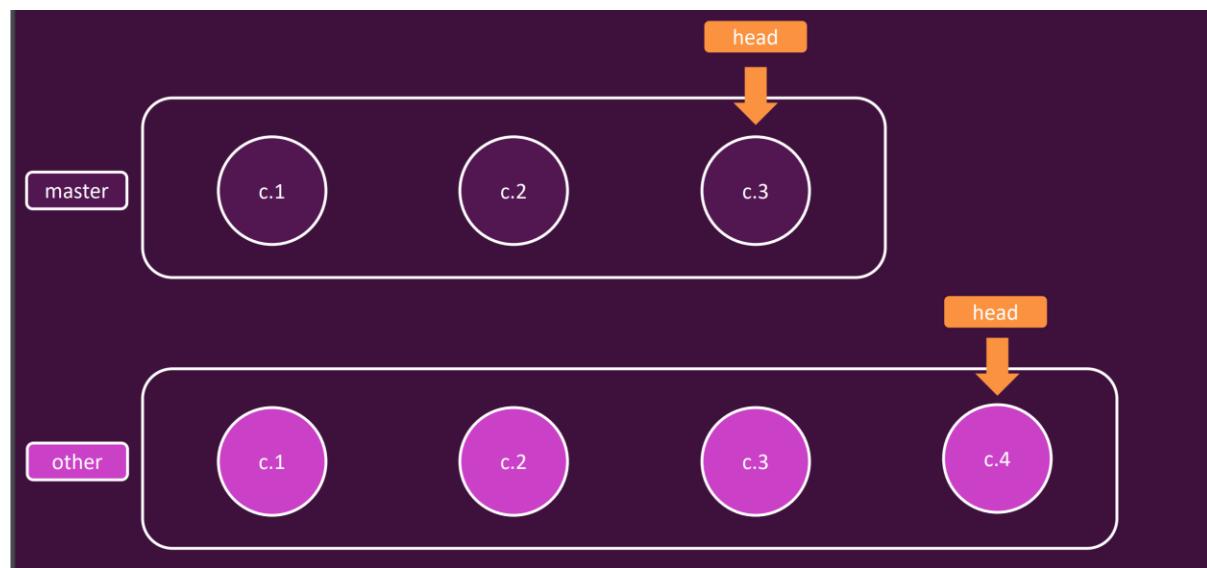


Fig. Head Understanding

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

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** Working_with_Branches.txt - git-basics - Visual Studio Code.
- Explorer:** Shows two files: initial-commit.txt and second-commit.txt.
- Terminal:** The terminal tab is selected. A red box highlights the "Wo" icon in the status bar.
- Terminal Content:**

```
D:\one drive data\Desktop\git-basics>git branch
  Landing-Page
* 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
```
- Status Bar:** Shows "cmd" and icons for close, minimize, maximize, and refresh.
- Bottom Status:** Shows "No symbols found in document 'Working_with_Branches.txt'".

Fig. Head Understanding History

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

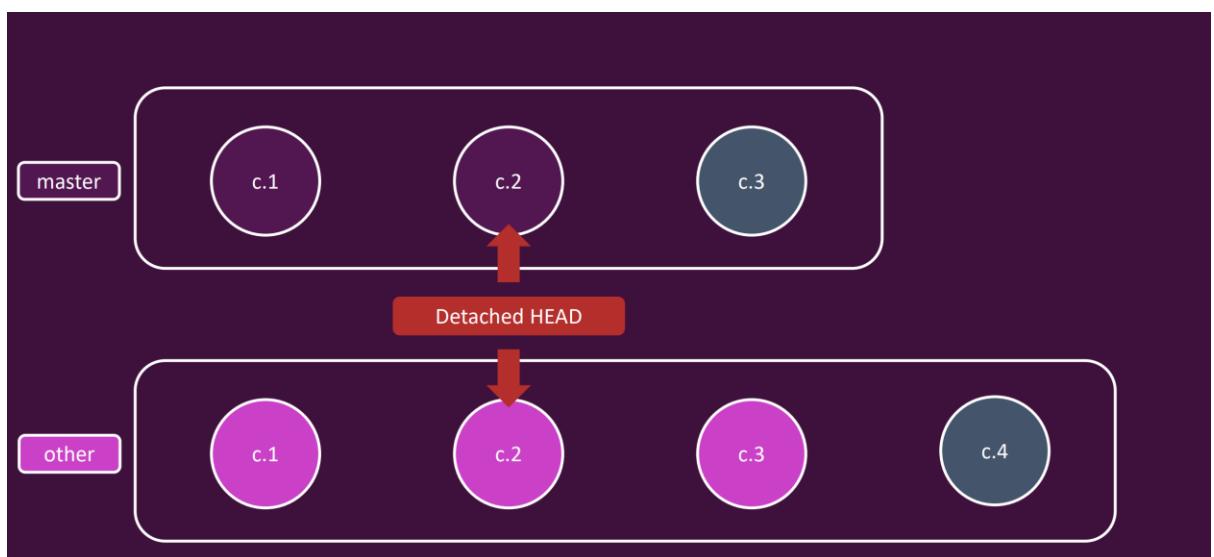


Fig. Detached Head

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 has a grey background for the detached head message and the 'git branch' output. The status bar at the bottom shows the path 'D:\one drive data\Desktop\git-basics' and the status '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 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
* (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>[]
```

The 'EXPLORER' sidebar shows a folder named 'GIT-BASICS' containing 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''. The 'TIMELINE' sidebar shows two entries: 'Branches are Working Saif P... 2 hrs' and 'File Renamed'. The bottom status bar shows 'Ln 1, Col 30 Spaces: 4 UTF-8 CRLF Plain Text'.

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.

A screenshot of the Visual Studio Code interface. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The title bar shows 'Working_with_Branches.txt - git-basics - Visual Studio Code'. The left sidebar has sections for EXPLORER (GIT-BASICS folder containing initial-commit.txt, second-commit.txt, and Working_with_Branches.txt), PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL. The TERMINAL tab is active, displaying 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 OUTLINE section below the terminal shows '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:

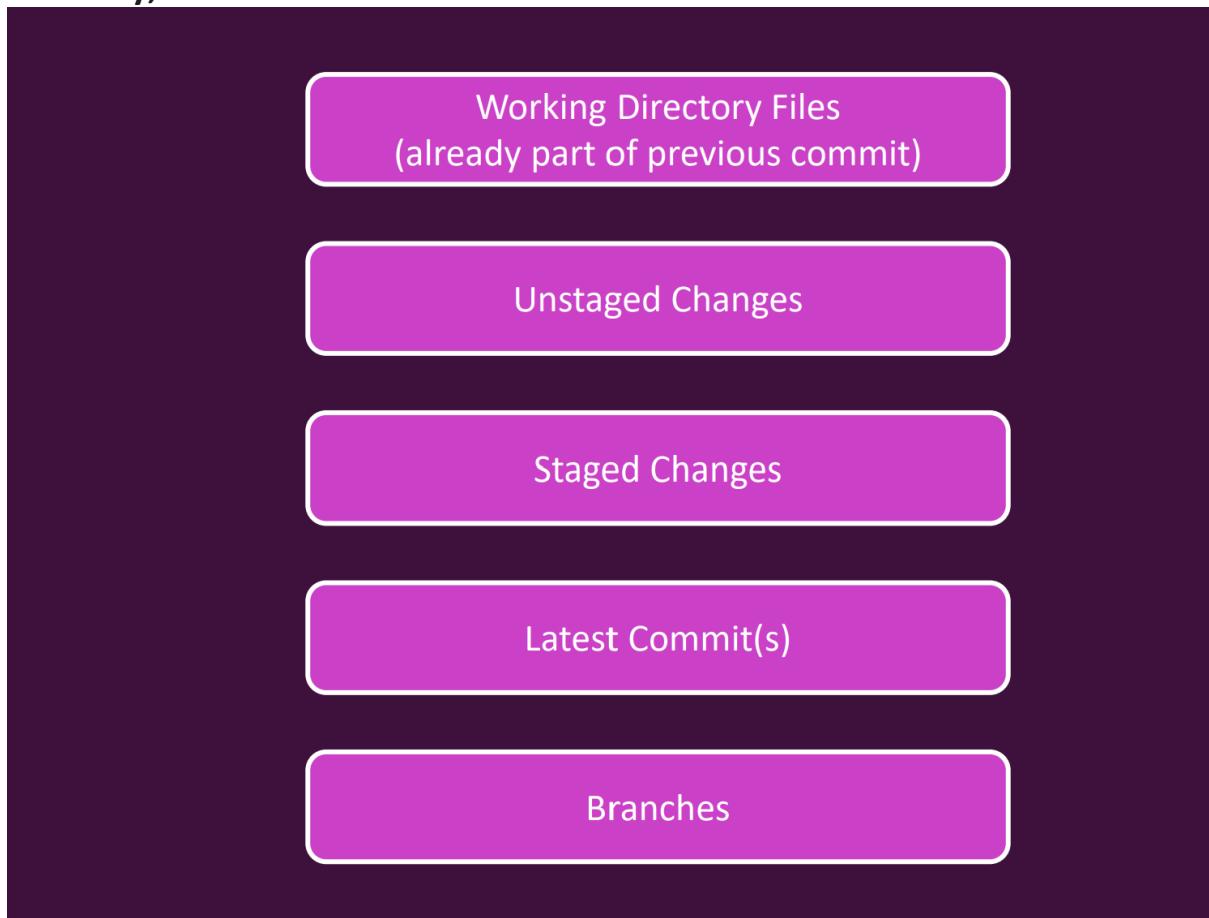
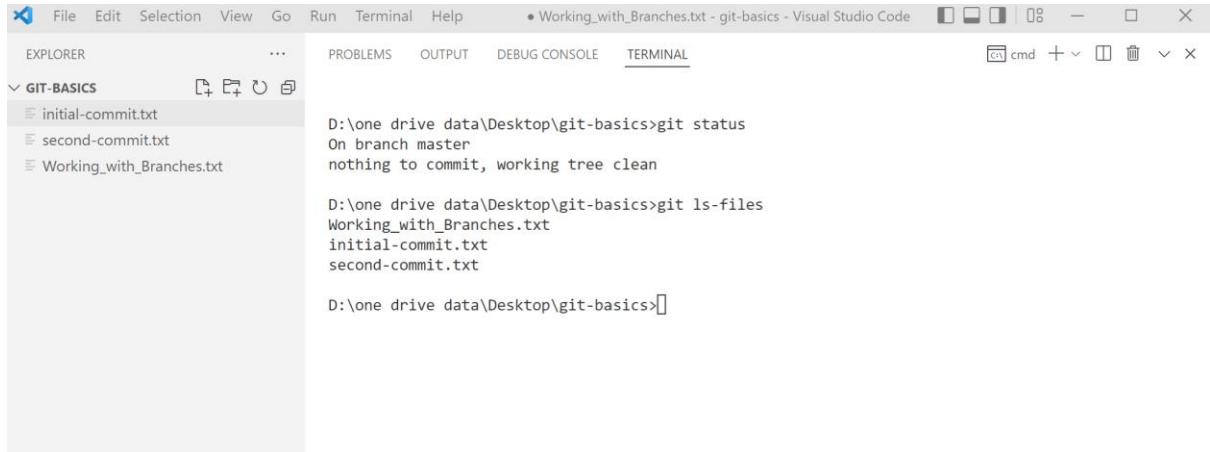


Fig. Deleting Data : Overview

Deleting Working Directory Files:

To check Which files currently on staging area.

git ls-files

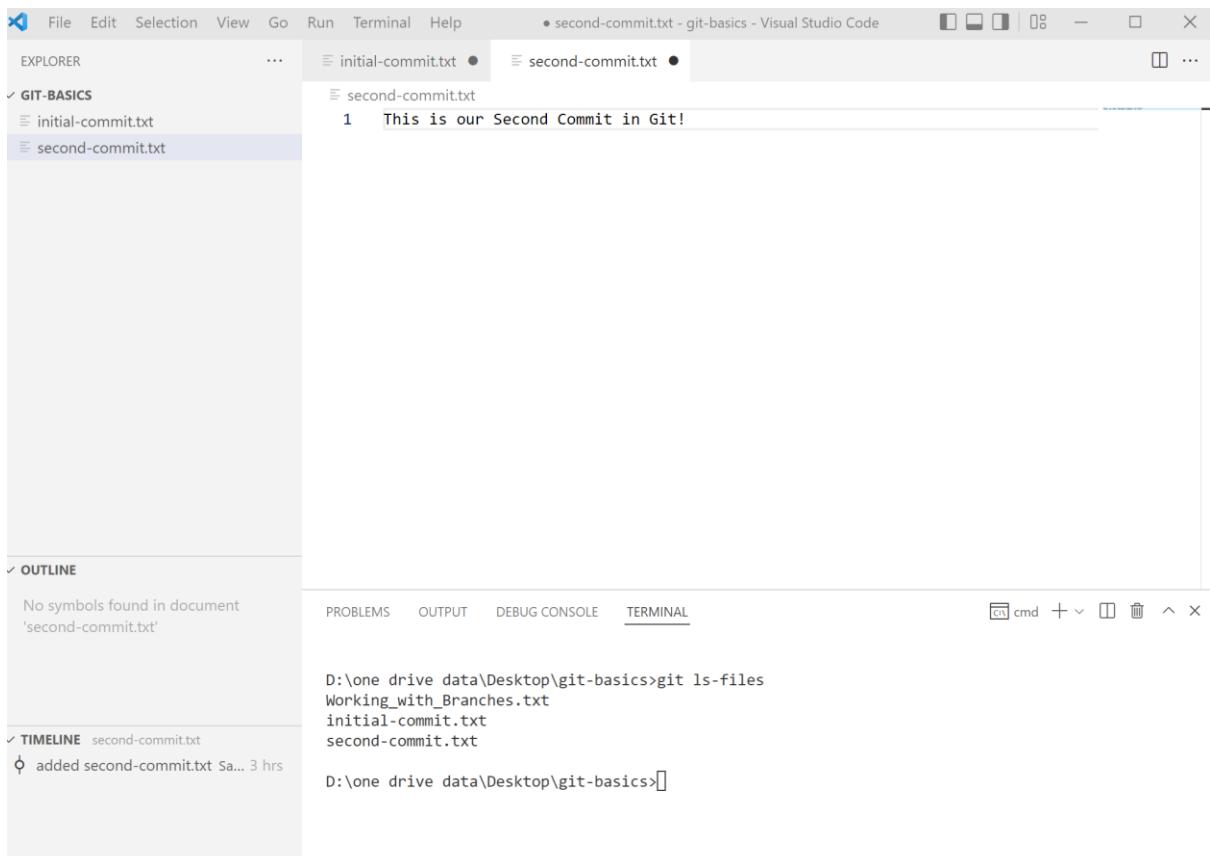


D:\one drive data\Desktop\git-basics>git status
On branch master
nothing to commit, working tree clean

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



initial-commit.txt ● second-commit.txt ●

1 This is our Second Commit in Git!

No symbols found in document 'second-commit.txt'

second-commit.txt

added second-commit.txt Sa... 3 hrs

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 the 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. In the Explorer sidebar, there are two files: 'initial-commit.txt' and 'second-commit.txt'. The 'initial-commit.txt' file is selected and has a modified status indicator ('M'). In the main editor area, the file contains:

```

1 Our First Step in git!
2
3 The text is no longer needed!

```

In the bottom right corner of the editor, there is a status bar message: 'modified: initial-commit.txt'. The terminal tab is active, showing the command 'git status' and its 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>

```

The status bar at the bottom of the terminal window also shows: 'Ln 3, Col 30 Spaces: 4 UTF-8 CRLF Plain Text'.

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

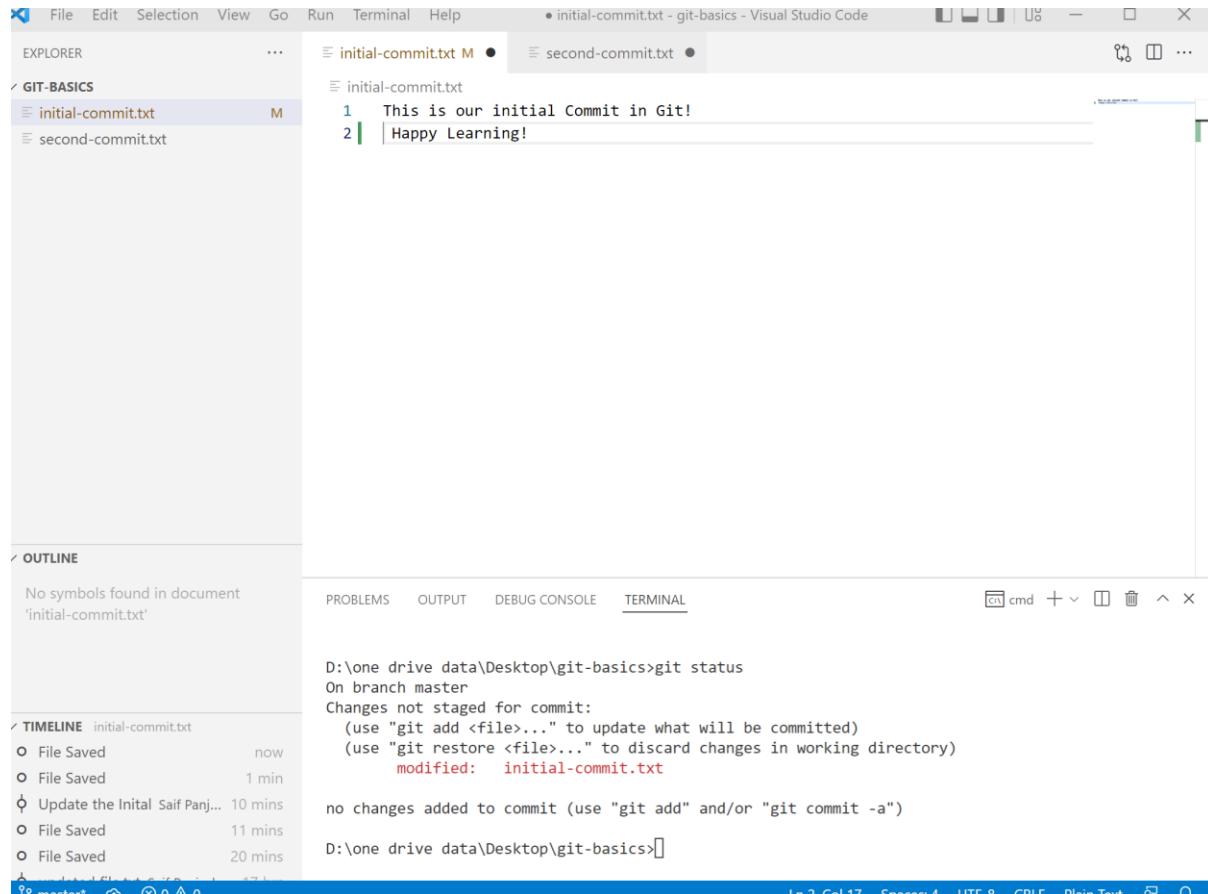
D:\one drive data\Desktop\git-basics>

```

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. The code editor shows the content of 'initial-commit.txt':

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

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

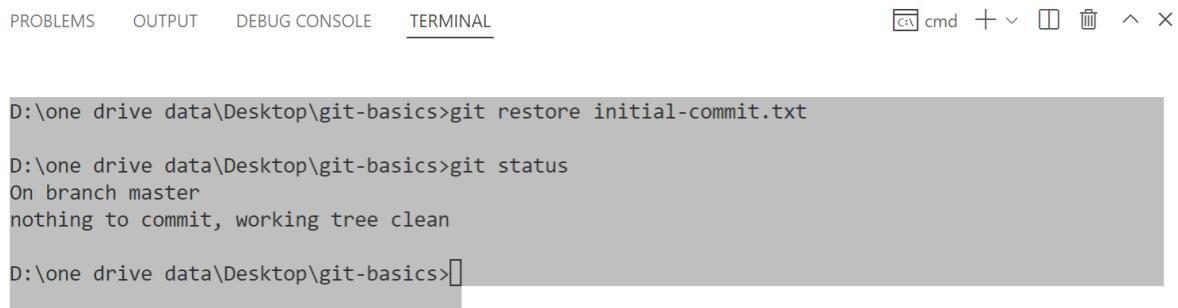
- File Saved now
- File Saved 1 min
- Update the Initial Saif Panj... 10 mins
- File Saved 11 mins
- File Saved 20 mins

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 a terminal window with the following command and output:

```
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

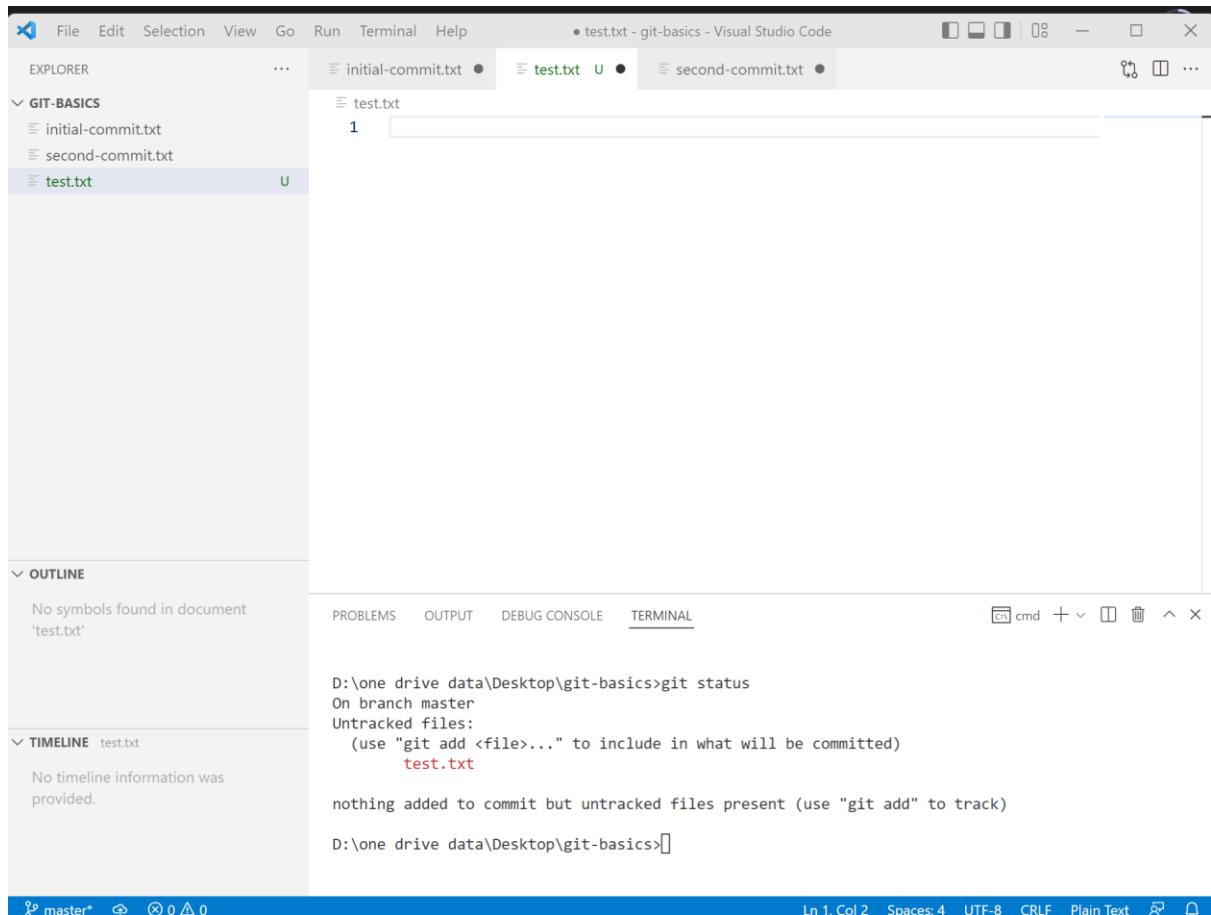
The screenshot shows the Visual Studio Code interface with the following details:

- 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".
- Terminal:** 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>[]
- Bottom Status Bar:** In 2 Col 1 Spaces: 4 LITE 8 CRLF 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. The Explorer sidebar on the left lists three files: 'initial-commit.txt', 'second-commit.txt', and 'test.txt'. The 'test.txt' file is selected and has a green 'U' icon next to it, indicating it is untracked. The terminal tab at the bottom shows the command line history:

```
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 indicates the file is in 'master' branch, has 0 changes, and is in 'Plain Text' mode.

Fig Created file test.txt



The screenshot shows the Visual Studio Code interface. The Explorer sidebar lists 'initial-commit.txt', 'second-commit.txt', and 'test.txt'. The 'test.txt' file is selected and has a green 'U' icon next to it. The terminal tab at the bottom shows the command line history:

```
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)

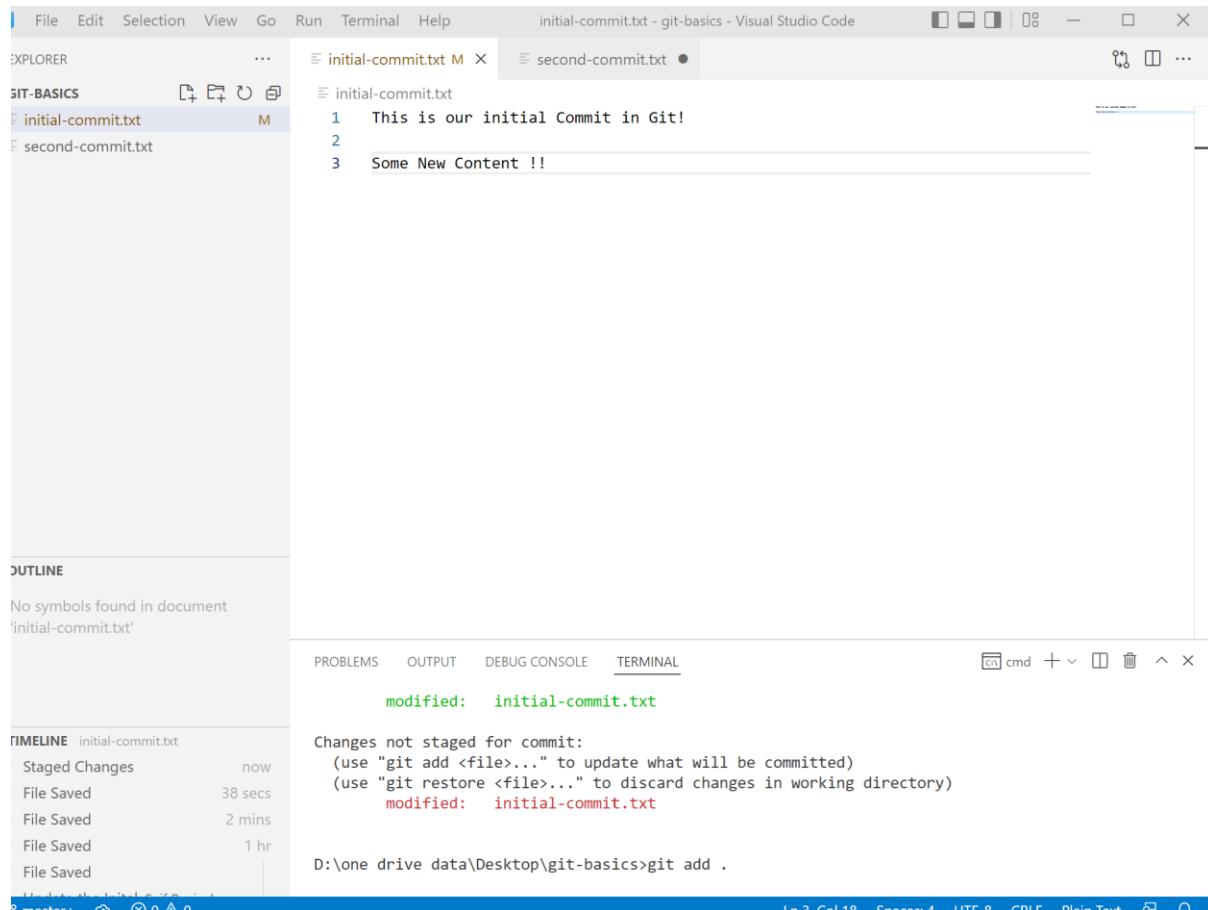
D:\one drive data\Desktop\git-basics>git clean -dn
Would remove test.txt

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:



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 and has a 'M' icon next to it, indicating it is modified. The code editor shows the following content:

```
1 This is our initial Commit in Git!
2
3 Some New Content !!
```

In the Timeline sidebar, under the heading 'initial-commit.txt', the following events are listed:

- Staged Changes now
- File Saved 38 secs
- File Saved 2 mins
- File Saved 1 hr
- File Saved

The Terminal tab is active, displaying the command line output:

```
modified: initial-commit.txt

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

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

Fig. Added Some Content

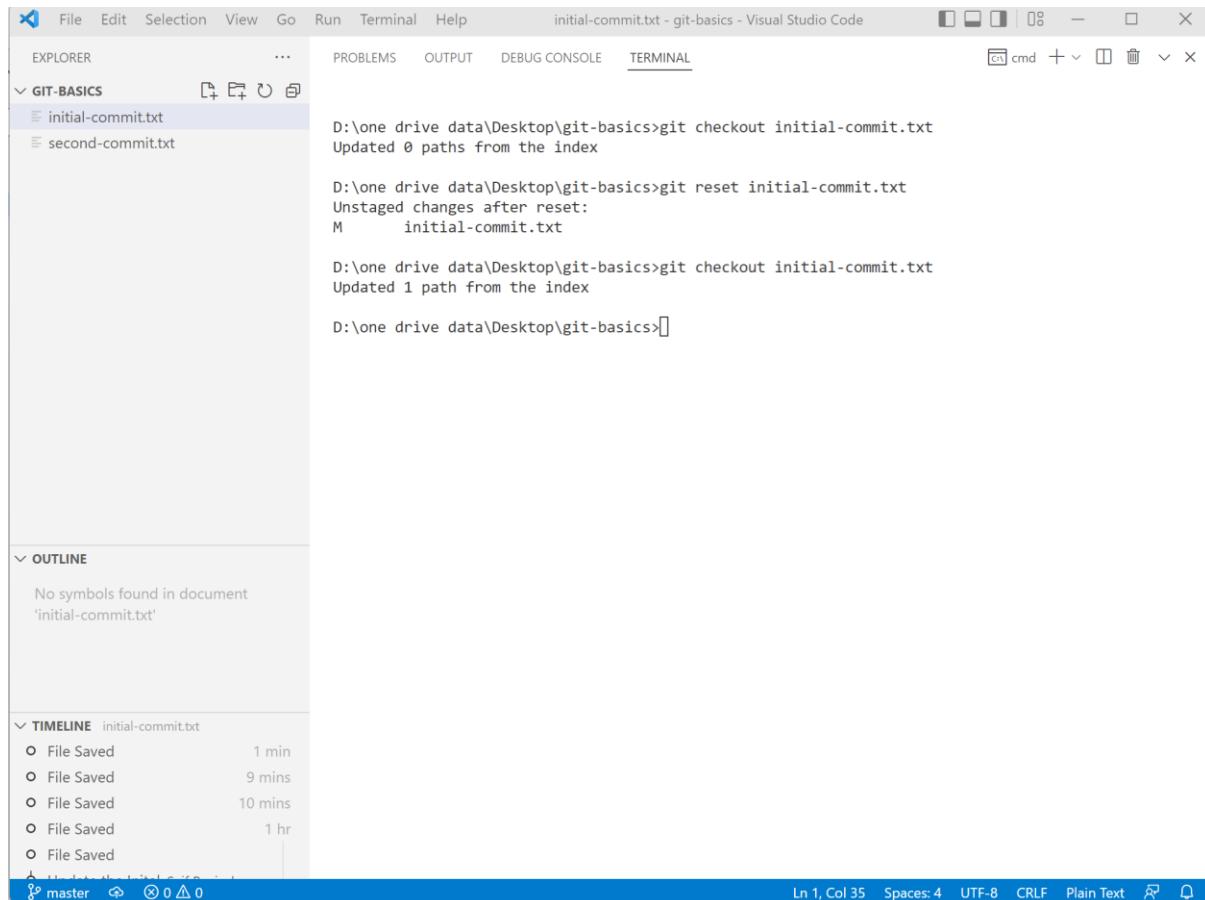
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
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:

- EXPLORER** view: Shows two files: `initial-commit.txt` and `second-commit.txt`.
- TERMINAL** tab: Displays the command-line history:
 - `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>[]`
- OUTLINE** view: Shows "No symbols found in document 'initial-commit.txt'".
- TIMELINE** view: Shows a list of file save events for `initial-commit.txt`:
 - File Saved (1 min ago)
 - File Saved (9 mins ago)
 - File Saved (10 mins ago)
 - File Saved (1 hr ago)
- Bottom status bar: Shows the current branch is `master`, 0 unstaged changes, and file statistics: 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

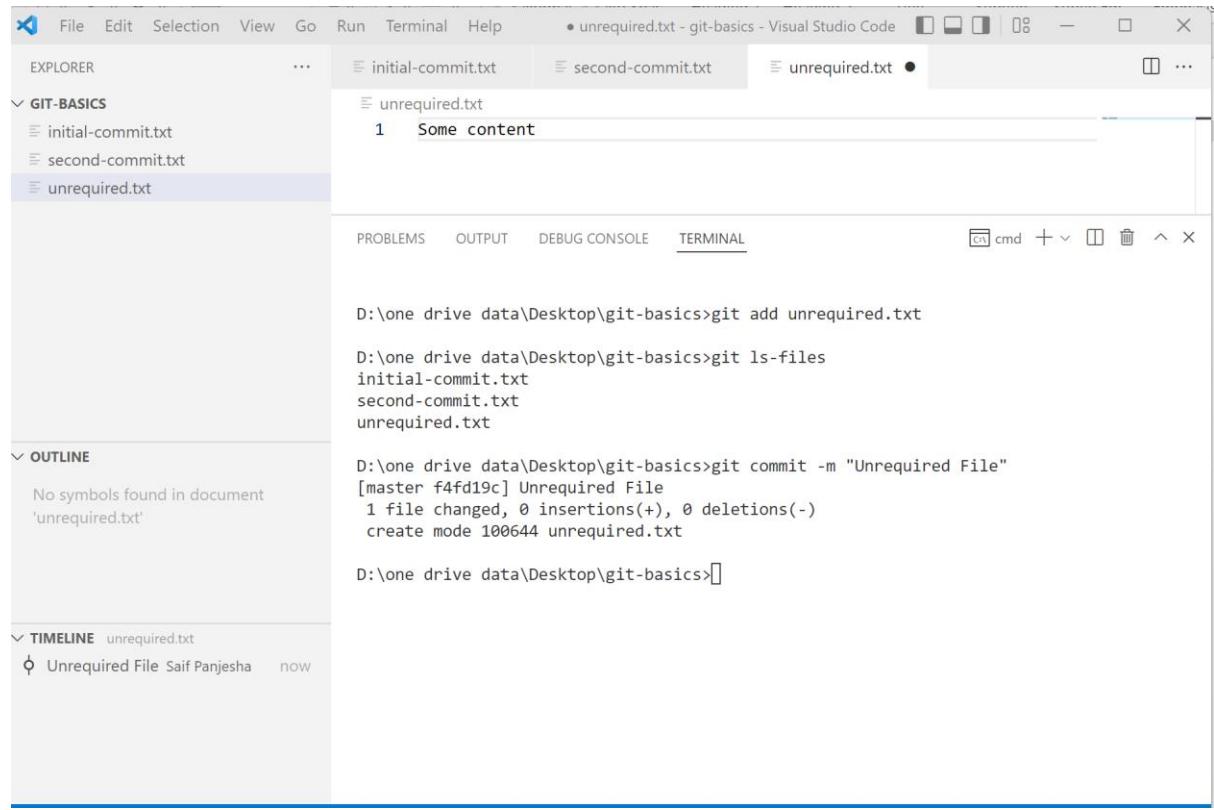
D:\one drive data\Desktop\git-basics>git restore --staged second-commit.txt
D:\one drive data\Desktop\git-basics>git checkout second-commit.txt
Updated 1 path from the index

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

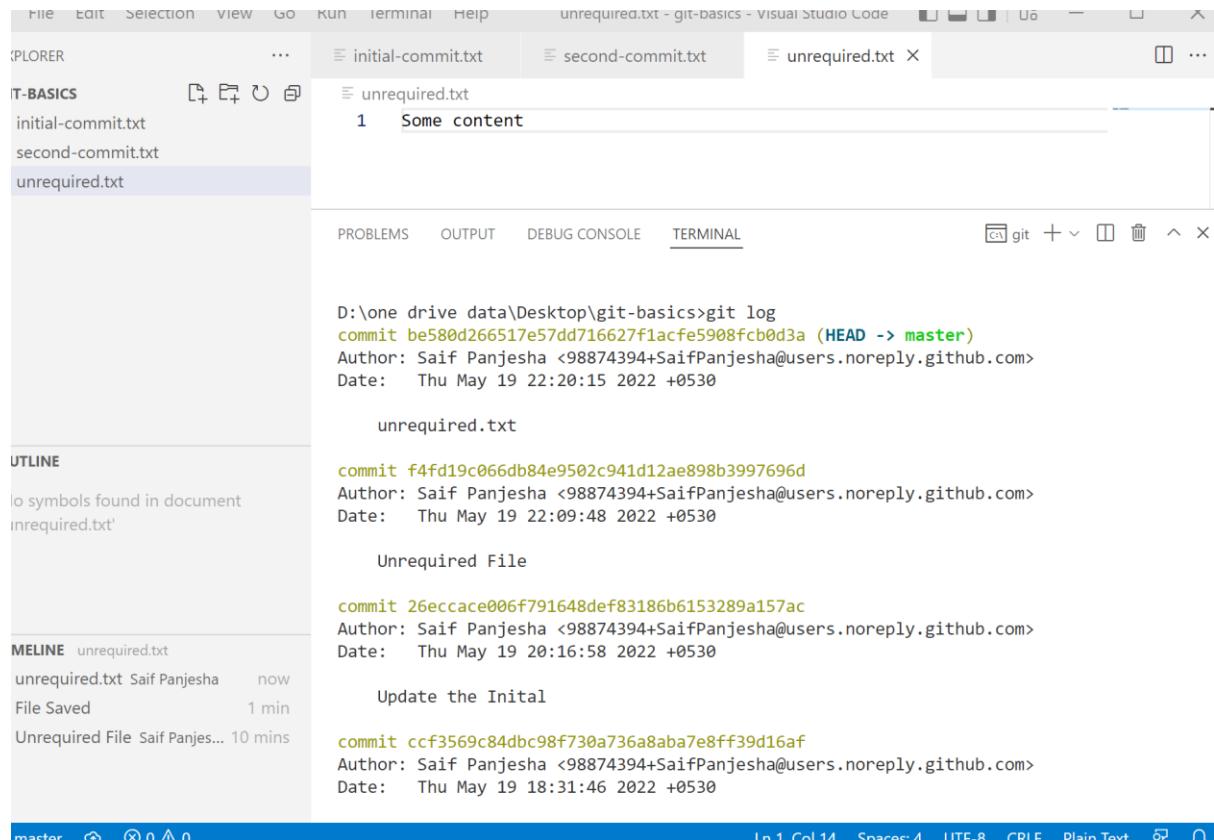
Deleting commit with git reset



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>

No symbols found in document 'unrequired.txt'

Timeline: unrequired.txt
Unrequired File Saif Panjesha now



```
D:\one drive data\Desktop\git-basics>git log  
commit be580d266517e57dd716627f1acfe5908fcbb0d3a (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 26eccace006f791648def83186b6153289a157ac  
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
```

File Saved 1 min
Unrequired File Saif Panjesha... 10 mins

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 selected. The terminal window displays the command "D:\one drive data\Desktop\git-basics>git reset --soft HEAD~2" followed by the log output of the repository. The log shows two 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 terminal also shows the status of the file "unrequired.txt" in the Explorer sidebar, which is listed under the "GIT-BASICS" section. The file "unrequired.txt" is highlighted in yellow, indicating it is the current file being worked on.

Fig. Head -> master to Update the Initial state

Default Command: git reset HEAD ~2

The screenshot shows the Visual Studio Code interface with the terminal tab selected. The terminal window displays the following command and its 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>
```

The Explorer sidebar shows three files: initial-commit.txt, second-commit.txt, and unrequired.txt. The Timeline sidebar 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:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Tab:** unrequired.txt - git-basics - Visual Studio Code
- Terminal:** Shows the command `git reset HEAD~2` and its output:
 - Unstaged changes after reset:
 - M initial-commit.txt
 - added second-commit.txt
 - commit 32072669068faa962f8e245df7e3be52c76accc
 - Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
 - Date: Thu May 19 02:36:24 2022 +0530
- Explorer:** Shows three files: initial-commit.txt, second-commit.txt, and unrequired.txt (highlighted).
- Outline:** Shows "No symbols found in document 'unrequired.txt'".
- Timeline:** Shows "File Saved" 12 mins ago.
- Bottom Status Bar:** master*, 0△0, Ln 1, Col 14, Spaces: 4, UTF-8, CRLF, Plain Text, icons for Find, Replace, and Save.

Fig. Success Deletion on unrequired file

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

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Tab:** unrequired.txt - git-basics - Visual Studio Code
- Terminal:** Shows the command `git ls-files` and its output:
 - initial-commit.txt
 - second-commit.txt
- Explorer:** Shows three files: initial-commit.txt, second-commit.txt, and unrequired.txt (highlighted). The unrequired.txt file contains the content "1 Some content".
- Bottom Status Bar:** master*, 0△0, Ln 1, Col 14, Spaces: 4, UTF-8, CRLF, Plain Text, icons for Find, Replace, and Save.

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 (content: "Some content").
- GIT-BASICS**: Shows files: initial-commit.txt, second-commit.txt.
- OUTLINE**: Shows: No symbols found in document 'unrequired.txt'.
- TIMELINE**: Shows: File Saved 24 mins ago.
- TERMINAL**: Shows command output:

```
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**: Shows: No symbols found in document 'unrequired.txt'.
- TIMELINE**: Shows: File Saved 24 mins ago.
- TERMINAL**: Shows command output:

```
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>[]
```

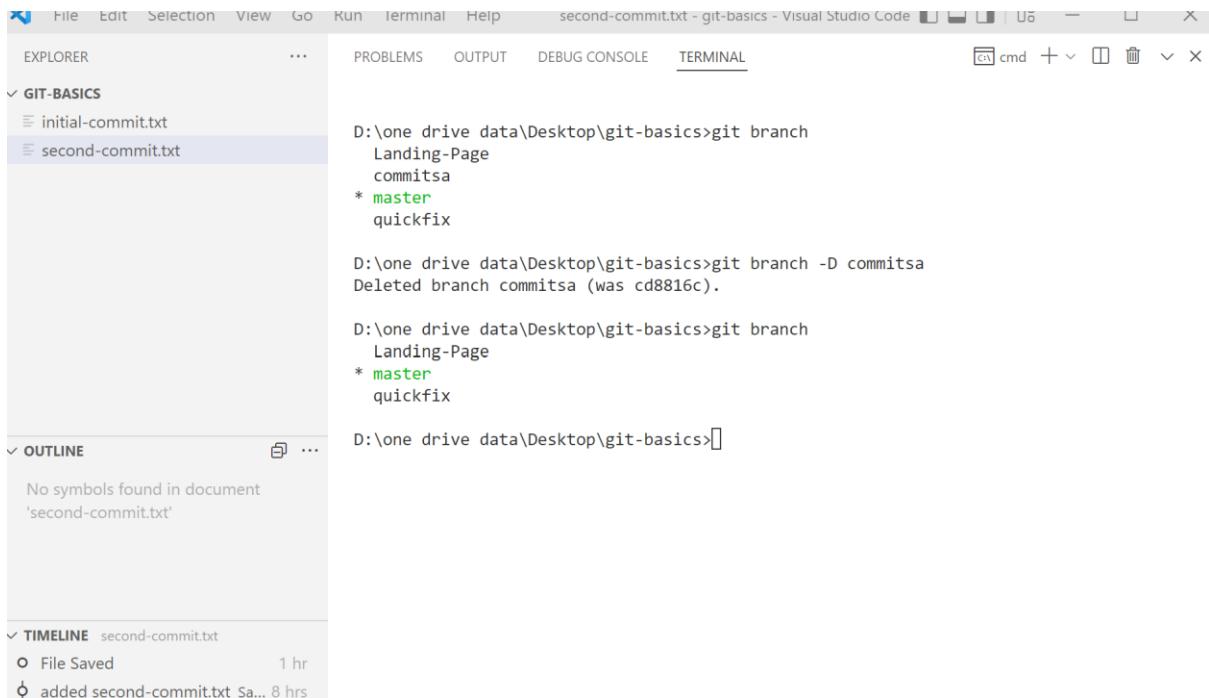
Fig. Head -> master to added second-commit.txt

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 in the editor. The terminal tab is active, displaying the following command-line session:

```
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>git branch
  Landing-Page
* master
  quickfix

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

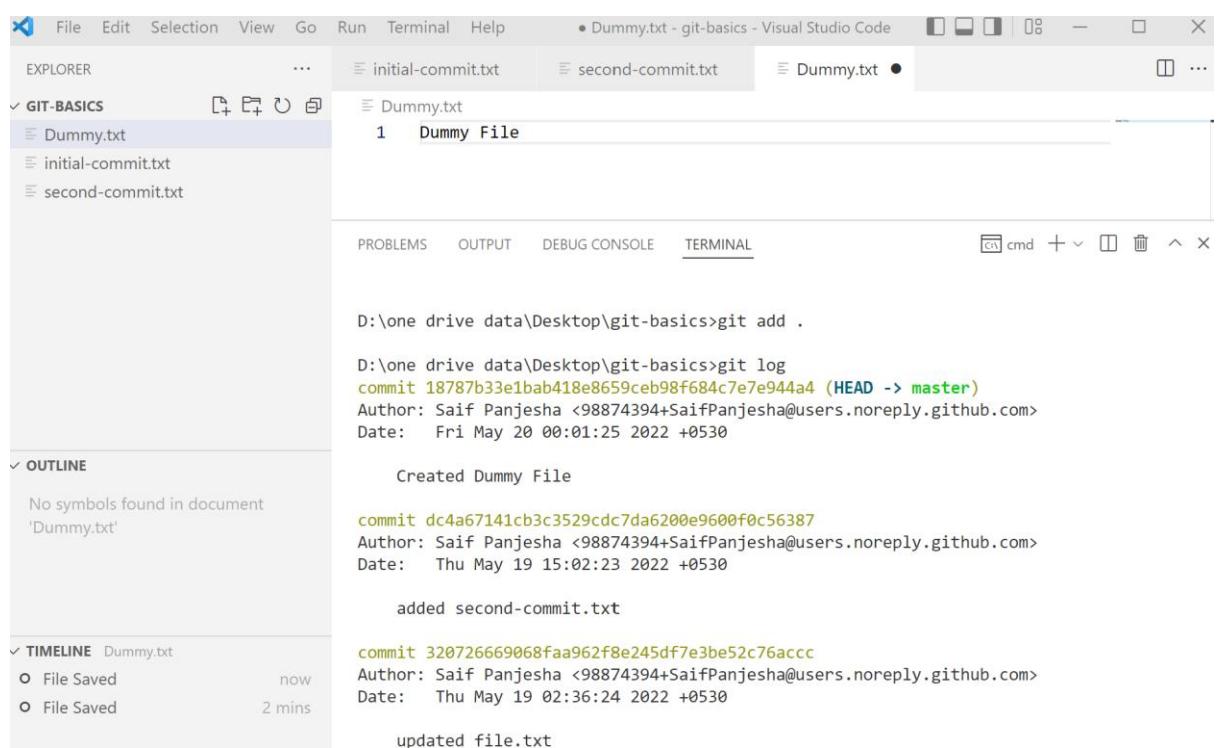
commit 320726669068faa962f8e245df7e3be52c76accc
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Thu May 19 02:36:24 2022 +0530

    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 '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

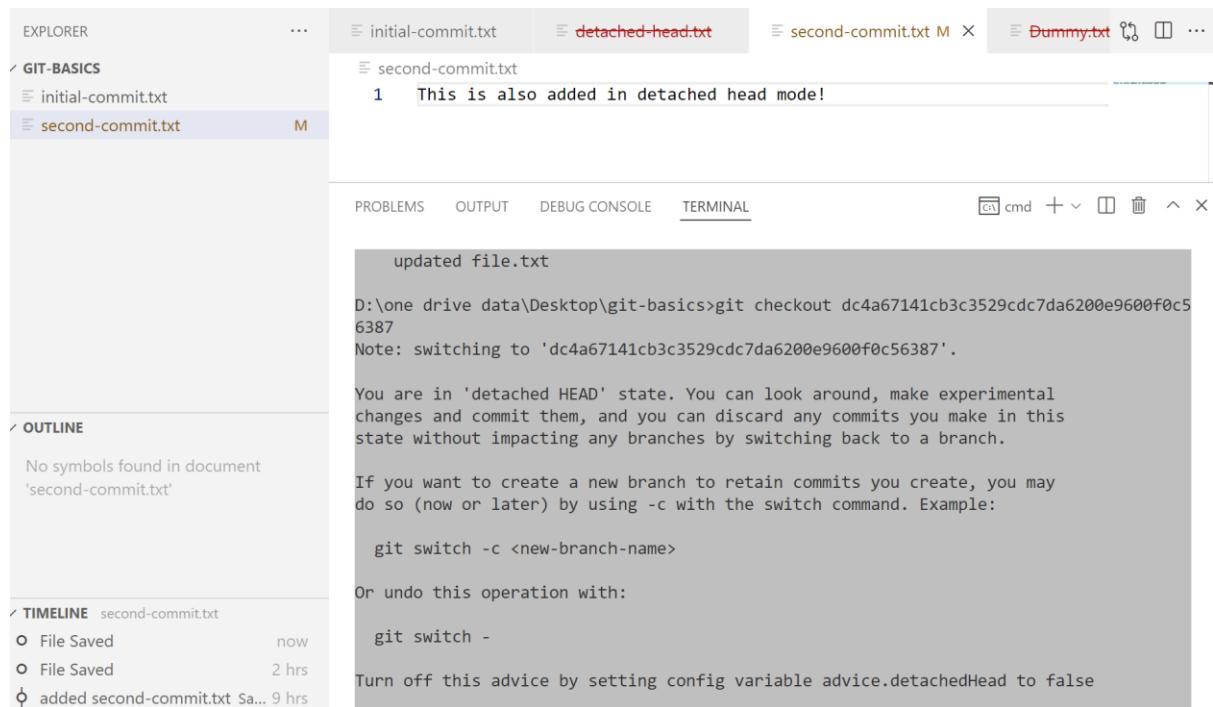


Fig. Added Some in Detached Head

```
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(+)
```

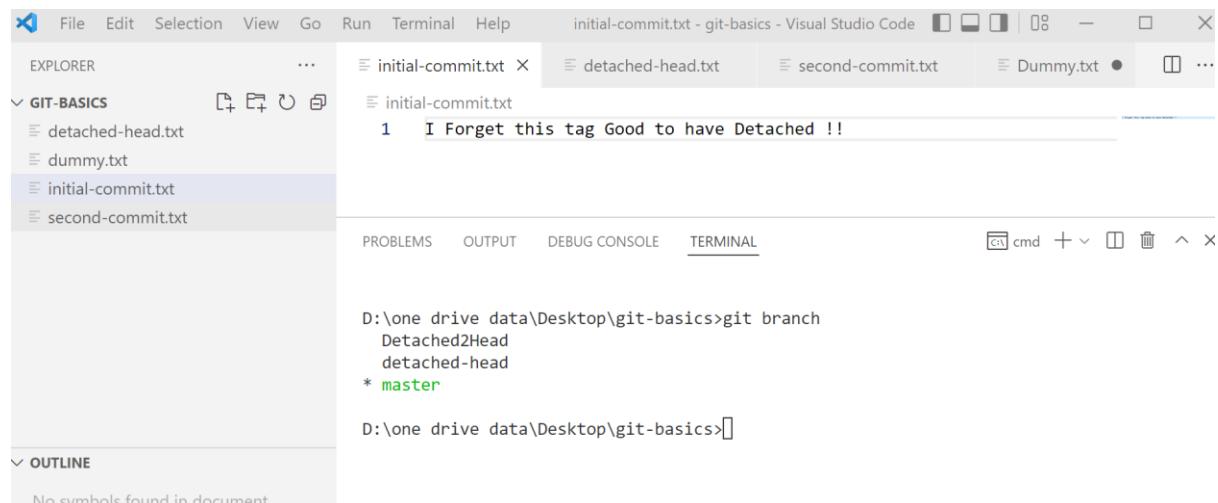


Fig. Succesful head Changes

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

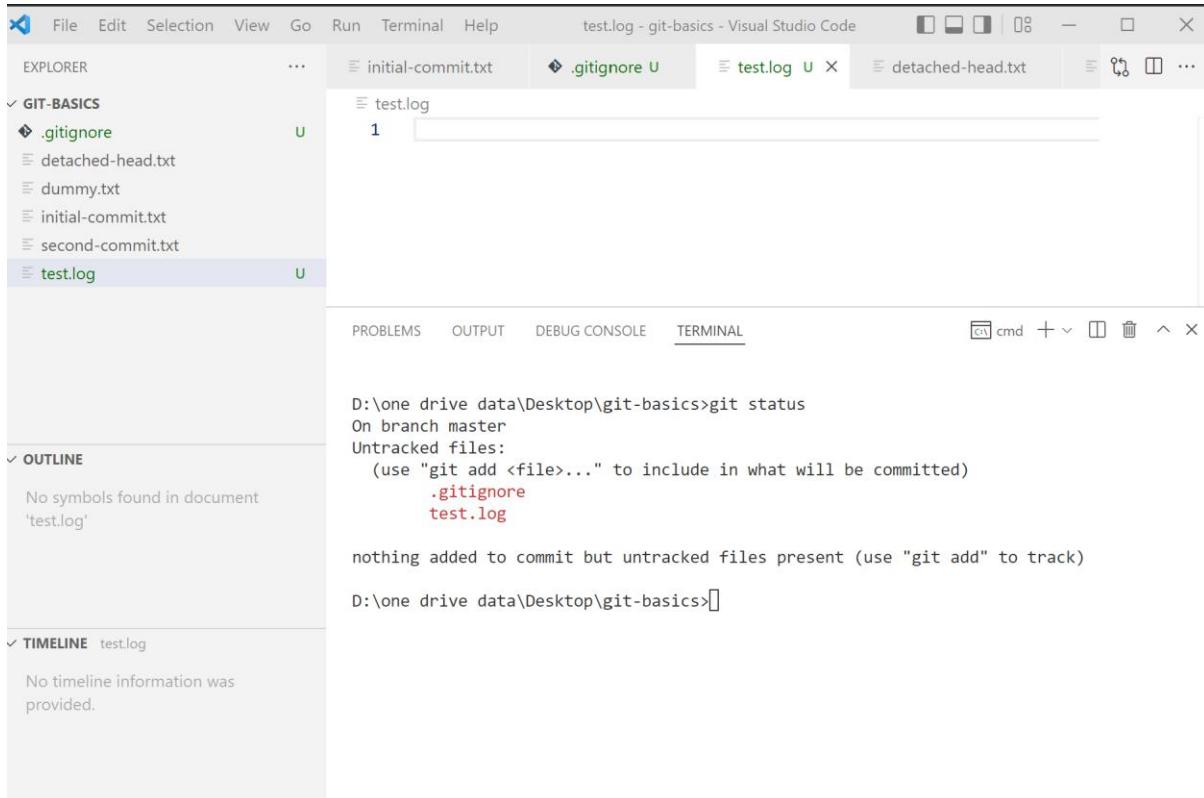
```
Deleted branch Detached2Head (was 4578010).
```

```
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. The Explorer sidebar on the left lists files and folders under 'GIT-BASICS'. The '.gitignore' file is open in the center editor, containing the line 'test.log'. The terminal tab at the bottom shows the command 'git status' being run in the directory 'D:\one drive data\Desktop\git-basics', which outputs that 'test.log' is an untracked file.

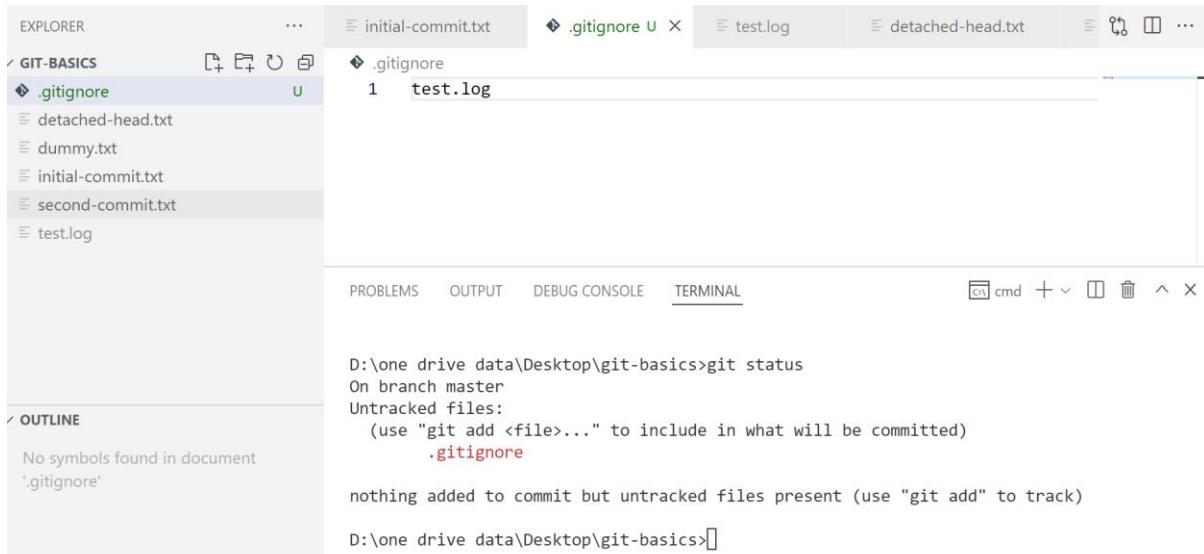
```
D:\one drive data\Desktop\git-basics>git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .gitignore
    test.log

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



This screenshot is similar to the previous one, showing the Visual Studio Code interface with the '.gitignore' file open. However, the 'test.log' entry has been removed from the file. The terminal output remains the same, indicating that 'test.log' is still an untracked file.

```
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

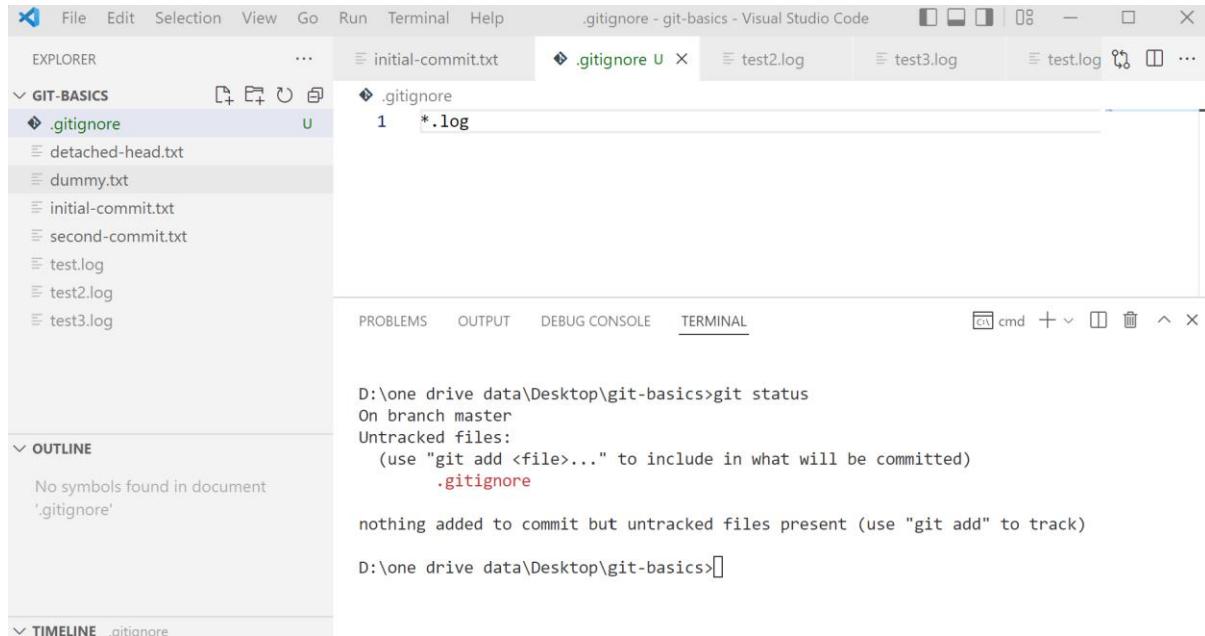
nothing added to commit but untracked files present (use "git add" to track)

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

Fig. test.log File ignored

To Ignore Multiple Files:

***.log – for ignorance all files**

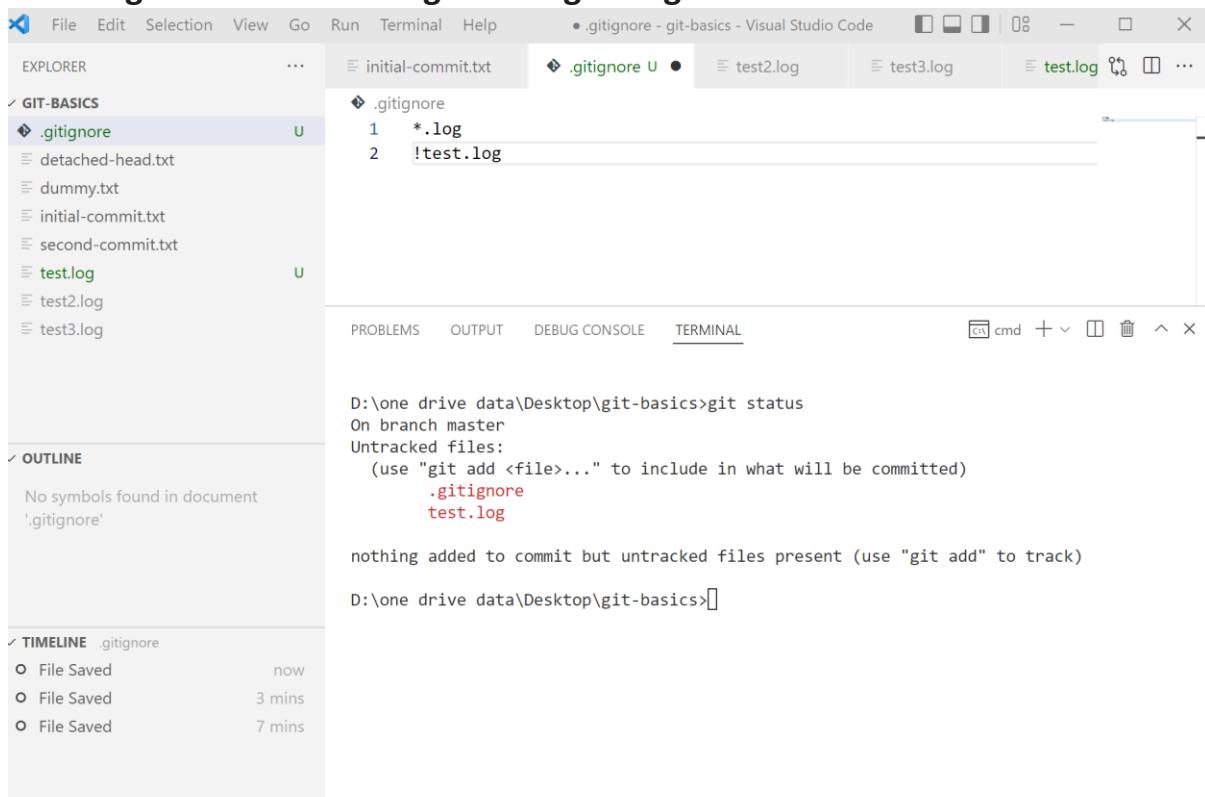


The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left lists several files: detached-head.txt, dummy.txt, initial-commit.txt, second-commit.txt, test.log, test2.log, and test3.log. The .gitignore file in the GIT-BASICS folder contains the line `1 *.*.log`. The TERMINAL tab at the bottom shows the command `git status` being run, which outputs:

```
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. Multiple File get ignored test, test2 and test3

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



The screenshot shows the Visual Studio Code interface. The Explorer sidebar lists the same files as before. The .gitignore file in the GIT-BASICS folder contains the lines `1 *.*.log` and `2 !test.log`. The TERMINAL tab shows the `git status` command 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)
    .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 with the following details:

- EXPLORER** sidebar: Shows a tree view of files and folders. The **GIT-BASICS** folder contains `index.html`, `app-data`, and `.gitignore`. The `.gitignore` file is open in the editor.
- EDITOR**: The content of the `.gitignore` file is:


```
1 *.log
2 !test.log
3 app-data/*
```
- TERMINAL**: The terminal window shows the output of the command `git status`.


```
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 with the following details:

- EXPLORER** sidebar: Shows a tree view of files and folders. The **GIT-BASICS** folder contains `index.html`, `app-data`, and `.gitignore`. The `.gitignore` file is open in the editor.
- EDITOR**: The content of the `.gitignore` file is:


```
1 *.log
2 !test.log
3
```
- TERMINAL**: The terminal window shows the output of the command `git status`.


```
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>
```
- TIMELINE**: A timeline entry for the `.gitignore` file is visible at the bottom of the interface.

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 the Visual Studio Code interface. In the Explorer sidebar, there is a folder named 'GIT-BASICS' containing several files: detached-head.txt, dummy.txt, initial-commit.txt, and second-commit.txt. In the center, the '.gitignore' file is open, showing the following content:

```
1 *.log
2 !test.log
3
```

In the bottom right corner, the Terminal tab is active, displaying 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>
```

Fig. Clean all files & folders

Wrap Up What We Learn:

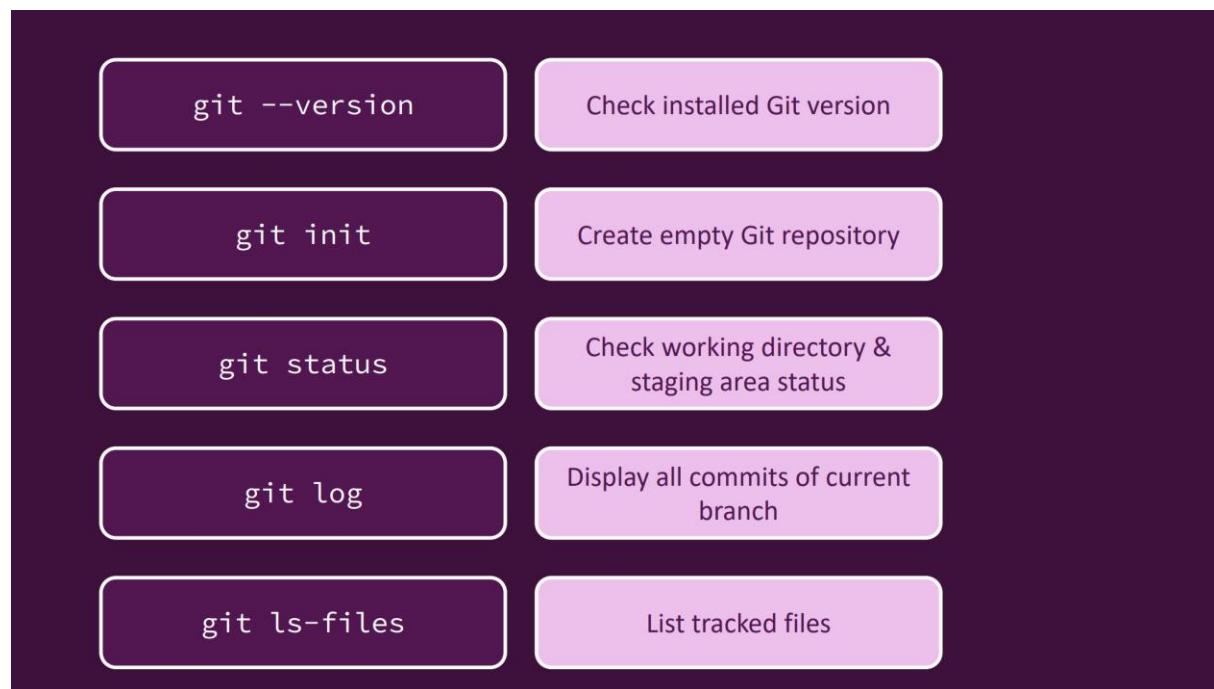


Fig . General Commands

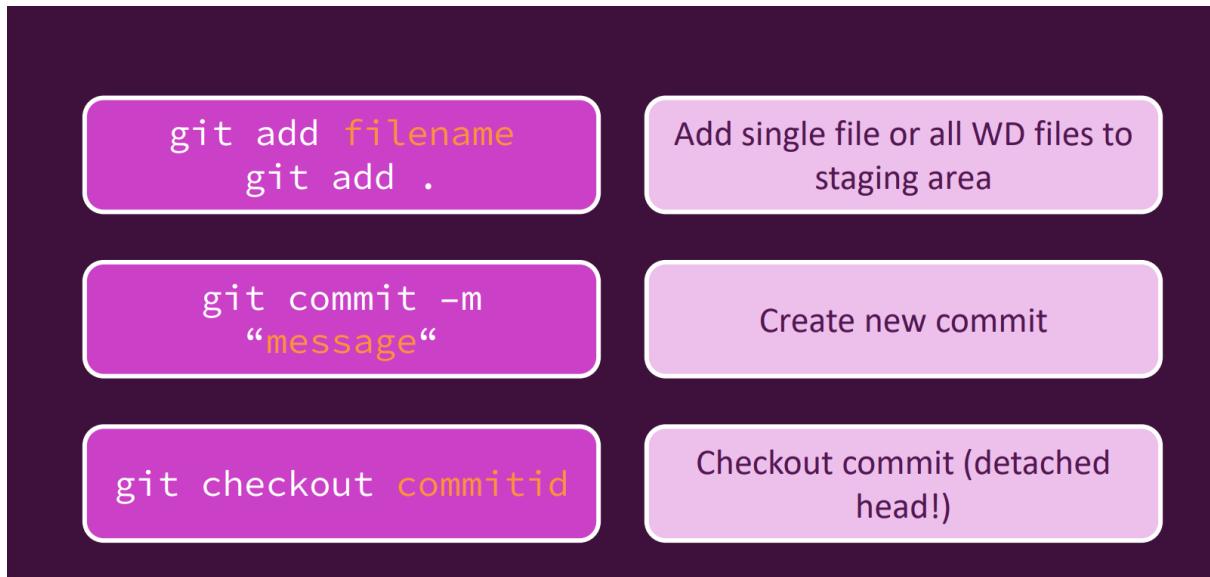


Fig. Basic Commit Creation and access

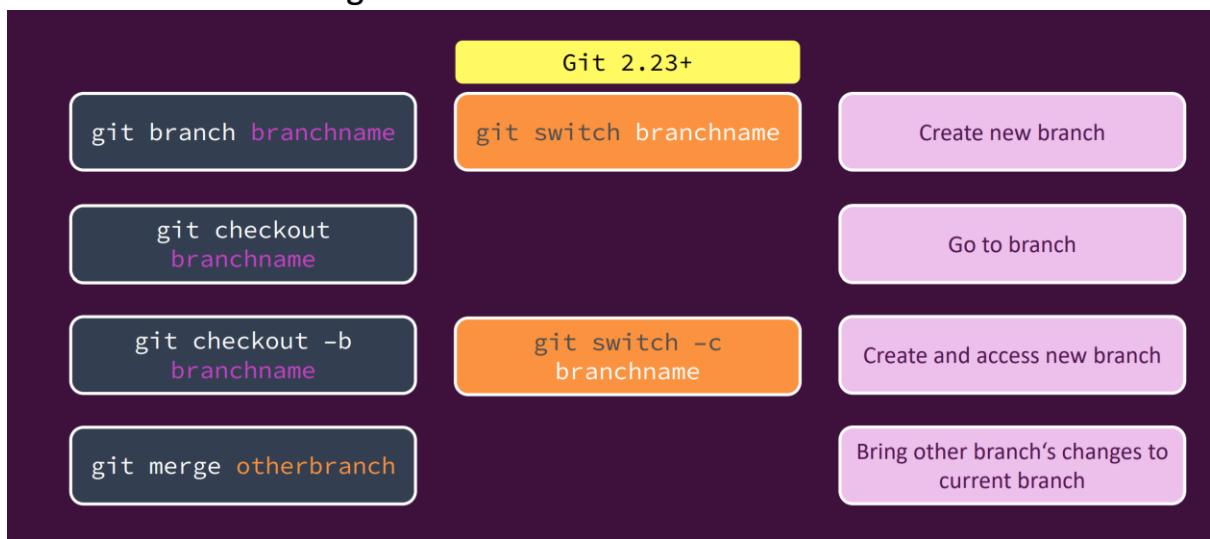


Fig. Branch and creation access

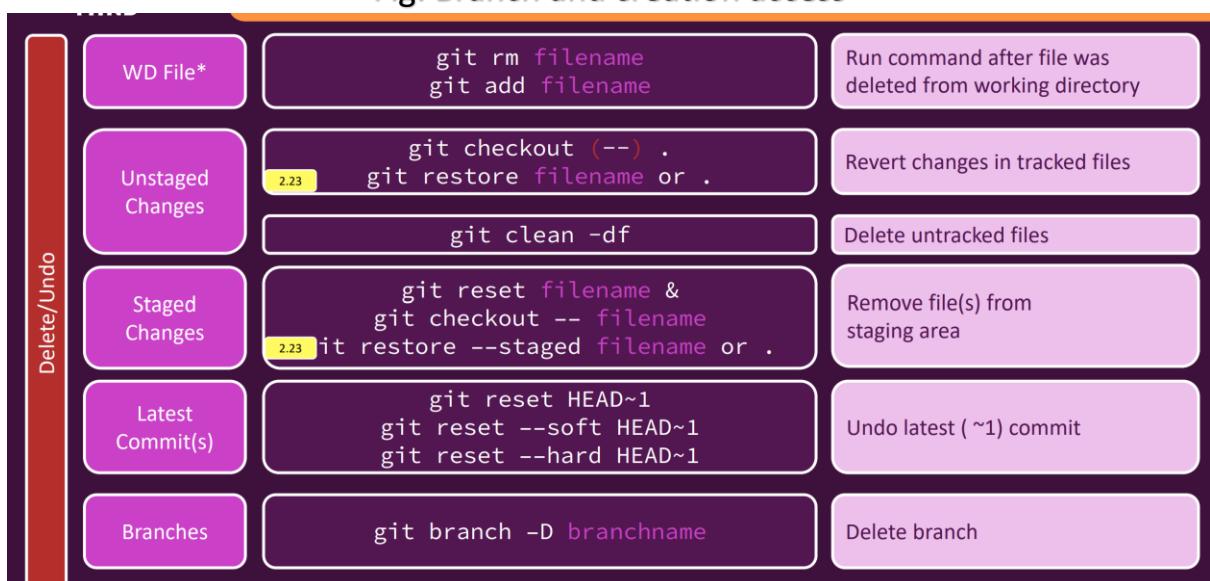


Fig. Deleting Data Summary

Diving Deep into Git



Fig. Module Introduction

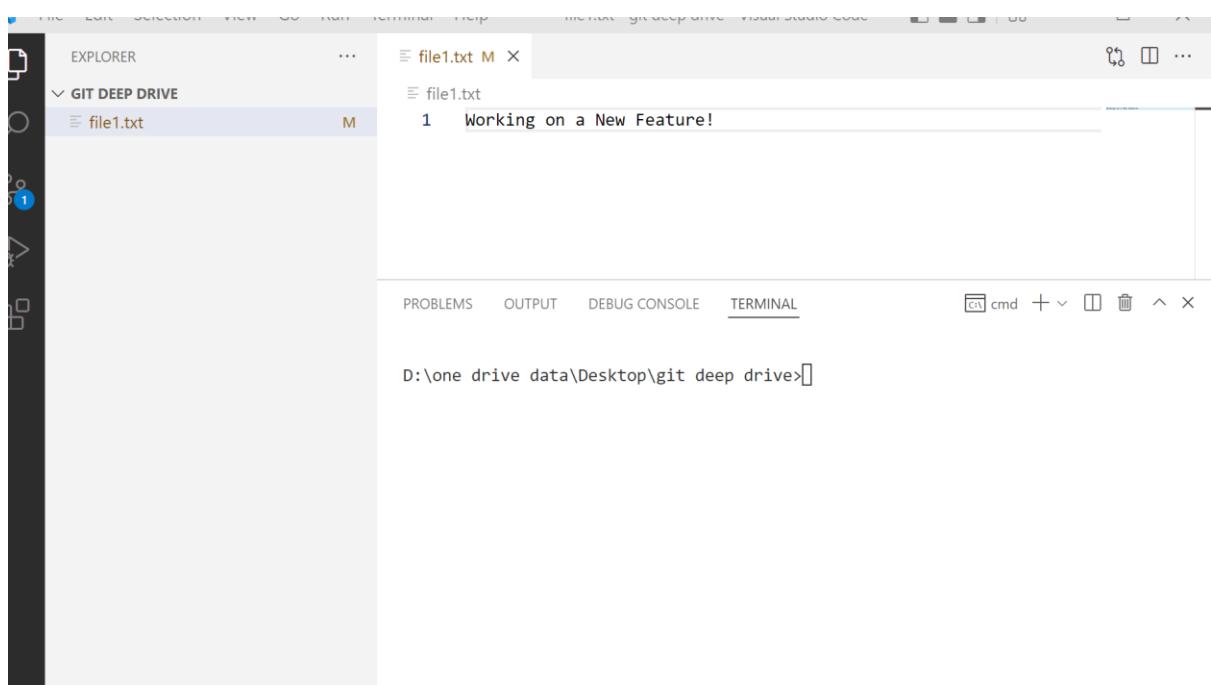
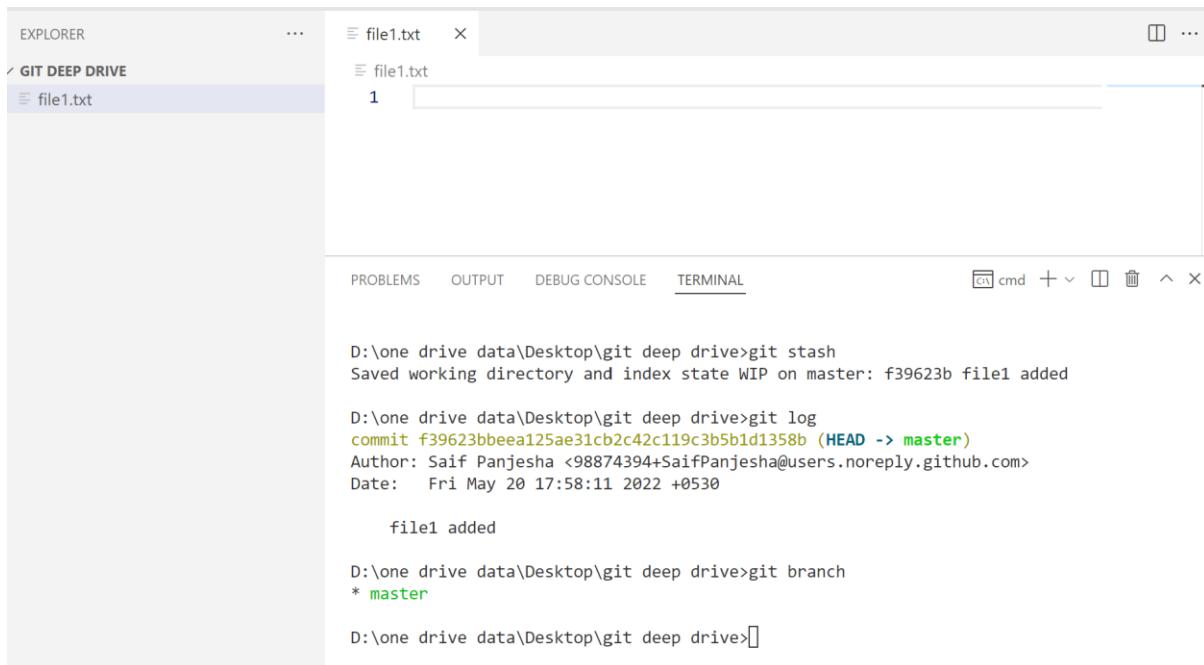


Fig. Created new file into Git Deep Drive Folder

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

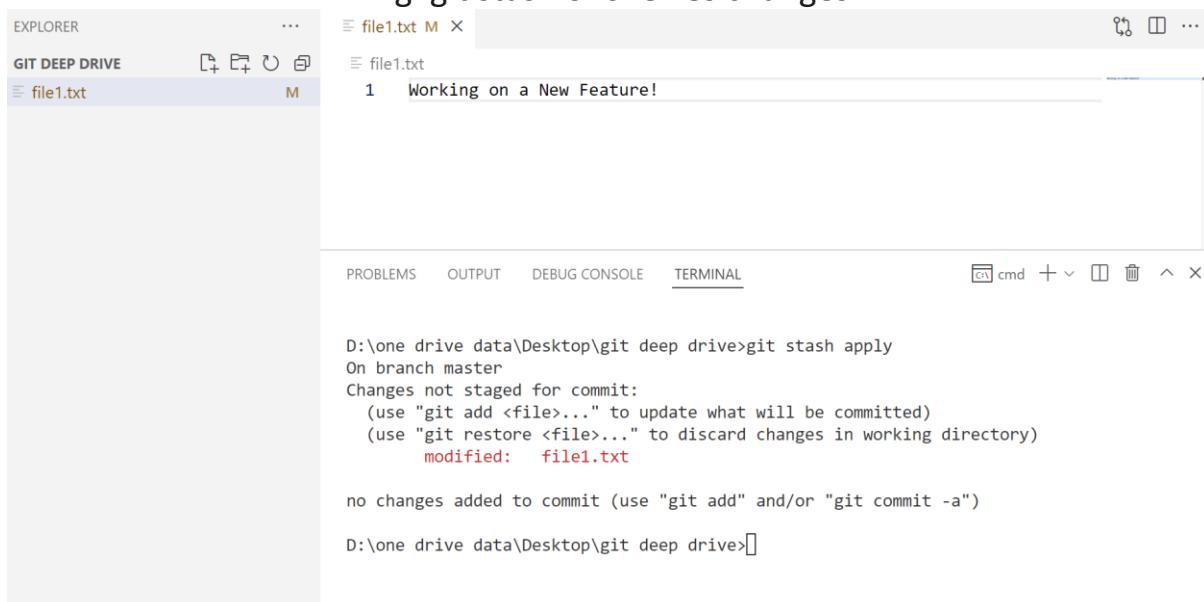
git stash: temporarily shelves (or stashes) change you've made to your working copy so you can work on something else, and then come back and re-apply them later on.



The screenshot shows the VS Code interface. In the Explorer sidebar, there's a folder named 'GIT DEEP DRIVE' containing a file 'file1.txt'. The terminal tab is active, displaying the following command history:

```
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. In the Explorer sidebar, there's a folder named 'GIT DEEP DRIVE' containing a file 'file1.txt'. The terminal tab is active, displaying the following command history:

```
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 .

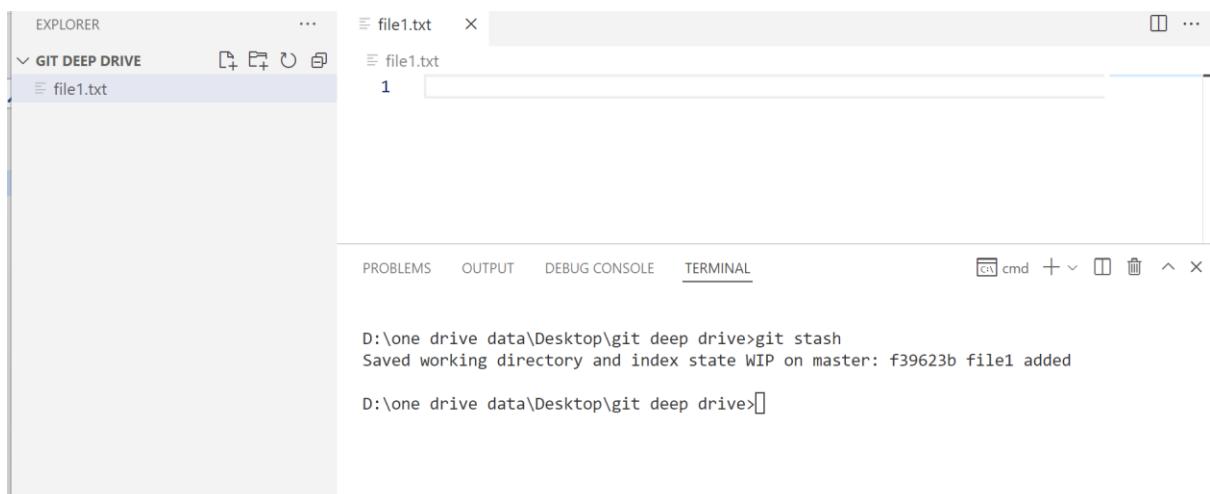


A screenshot of the Visual Studio Code interface. The Explorer sidebar shows a folder named 'GIT DEEP DRIVE' containing a file named 'file1.txt'. The main editor area displays the following text:

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

The terminal at the bottom shows the command line prompt 'D:\one drive data\Desktop\git deep drive>'.

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 area displays the number '1'.

The terminal at the bottom shows the command line prompt 'D:\one drive data\Desktop\git deep drive>'. The user then runs the command 'git stash', which outputs:

```
git stash
Saved working directory and index state WIP on master: f39623b file1 added
```

The terminal prompt then changes to 'D:\one drive data\Desktop\git deep drive>'.

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, applying the changes from the previous stash. The output shows the file `file1.txt` has been restored.

```
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 stashed changes with their respective IDs and details.

```
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. The Explorer sidebar shows a file named 'file1.txt'. The terminal window displays the following output:

```
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. The Explorer sidebar shows a file named 'file1.txt'. The terminal window displays the following 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 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 following details:

- EXPLORER**: Shows a folder named "GIT DEEP DRIVE" containing "file1.txt".
- TERMINAL**: Displays the command history and output of a git 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>
```
- PROBLEMS**, **OUTPUT**, **DEBUG CONSOLE**: These tabs are visible at the bottom of the terminal area.

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 following details:

- EXPLORER**: Shows a folder named "GIT DEEP DRIVE" containing "file1.txt".
- TERMINAL**: Displays the command history and output of a git 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")

D:\one\drive\data\Desktop\git\deep\drive>
```
- PROBLEMS**, **OUTPUT**, **DEBUG CONSOLE**: These tabs are visible at the bottom of the terminal area.

Fig. To Check our Latest changes with git stash apply

We can add message to our stash:

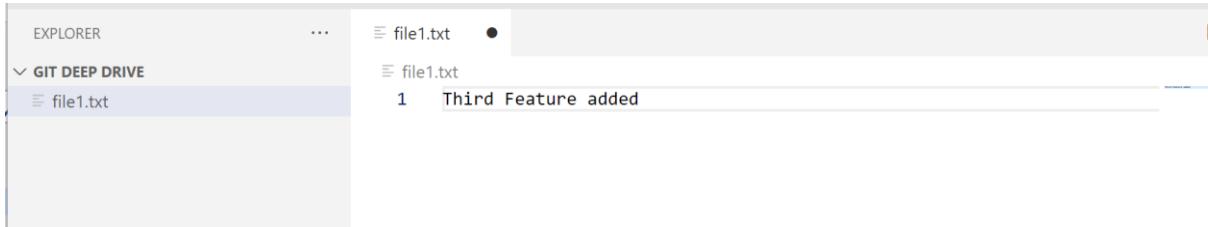


Fig. Third Feature added

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"' followed by its output, which shows the stash entry was saved. Below this, the command 'git stash list' is run, showing a list of six stash entries, each with a unique identifier, the commit hash, and the file name 'file1'.

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 VS Code interface with the following details:

- EXPLORER**: Shows a folder named "GIT DEEP DRIVE" containing "file1.txt".
- TERMINAL**: Displays the following command-line session:

```
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>
```

Terminal Output (Continued):

```
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
```

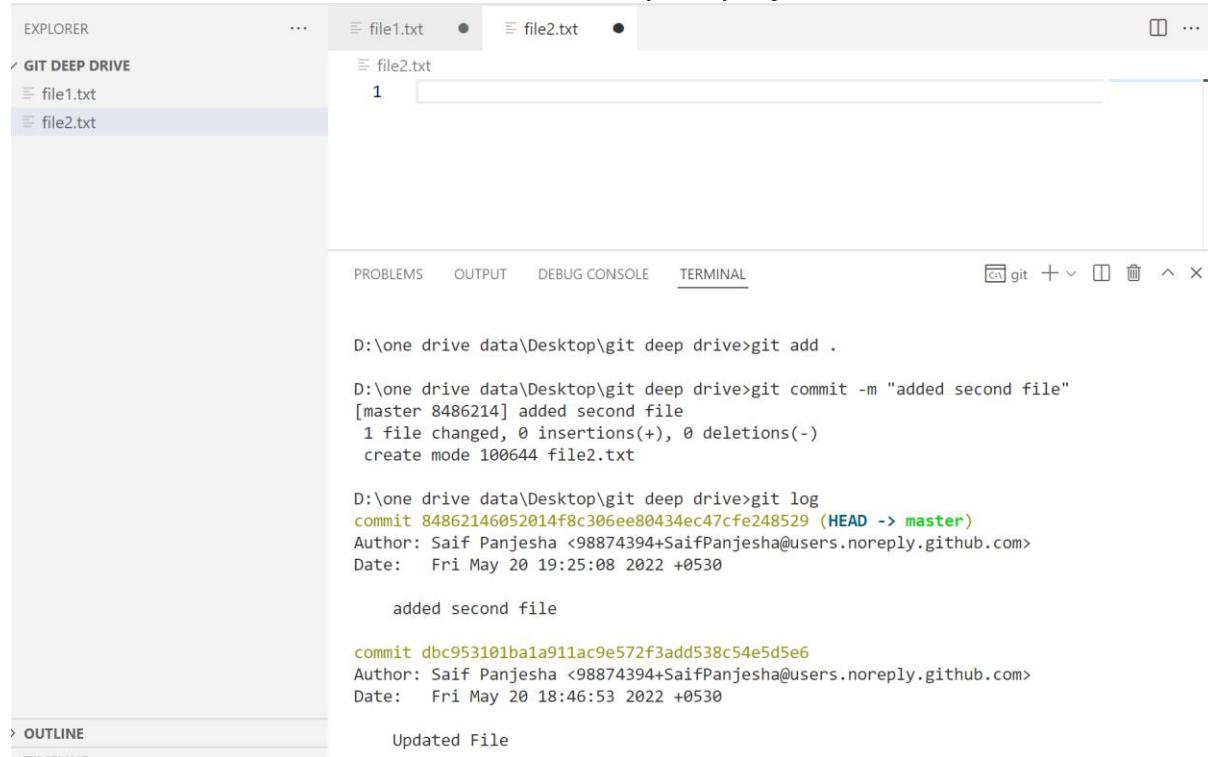
```
D:\one drive data\Desktop\git deep drive>git stash list
```

```
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 its content: "1". Below the editor, the terminal window displays 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

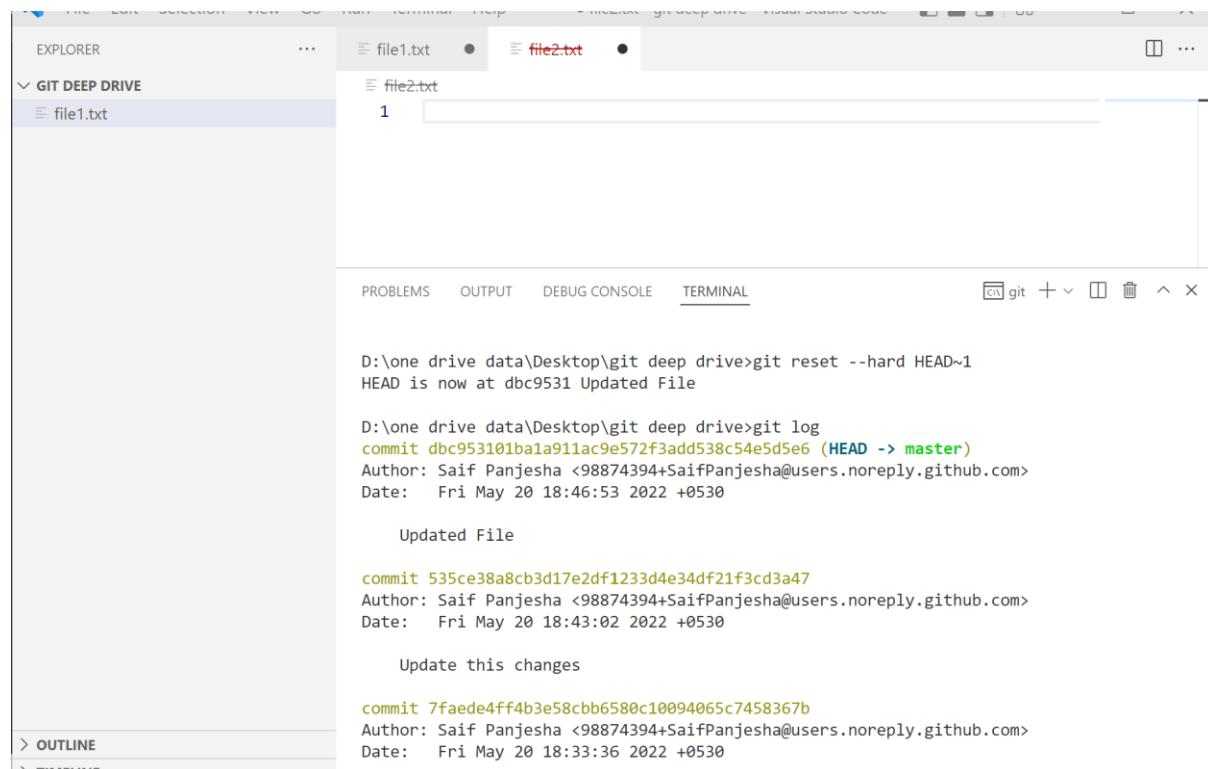
D:\one drive data\Desktop\git deep drive>git log
commit 84862146052014f8c306ee80434ec47cfe248529 (HEAD -> master)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 19:25:08 2022 +0530

    added second file

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

Below the terminal, the status bar indicates "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 its content: "1". Below the editor, the terminal window displays 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 dbc953101ba1a911ac9e572f3add538c54e5d5e6 (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 7faede4ff4b3e58ccb6580c10094065c7458367b
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 18:33:36 2022 +0530
```

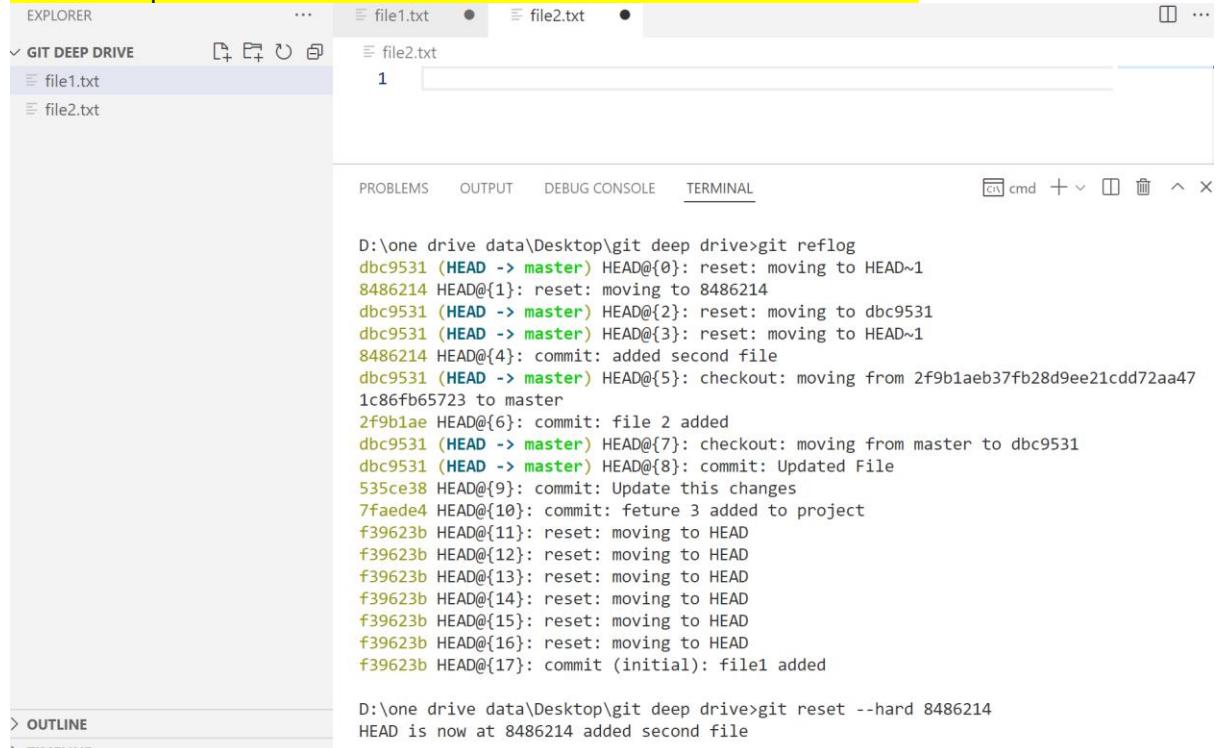
Below the terminal, the status bar indicates "Update this changes".

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."

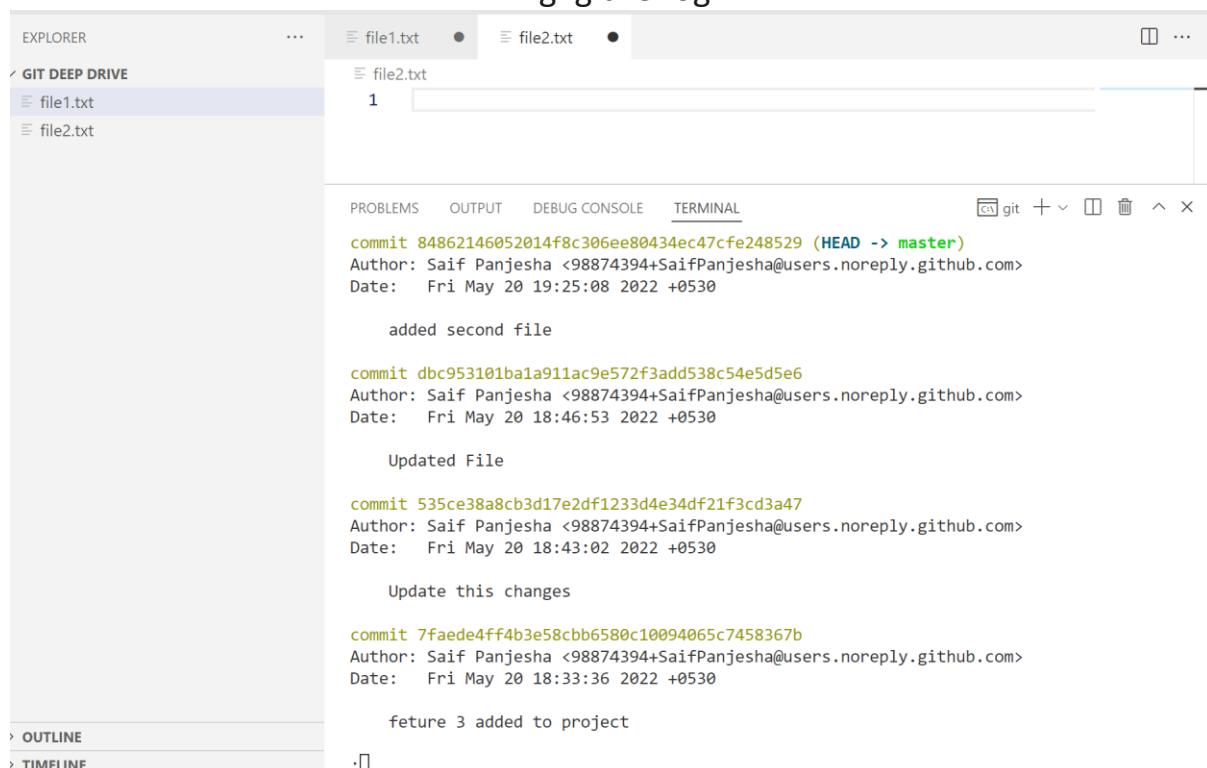
A branch tip is the last commit or most recent commit on a branch



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

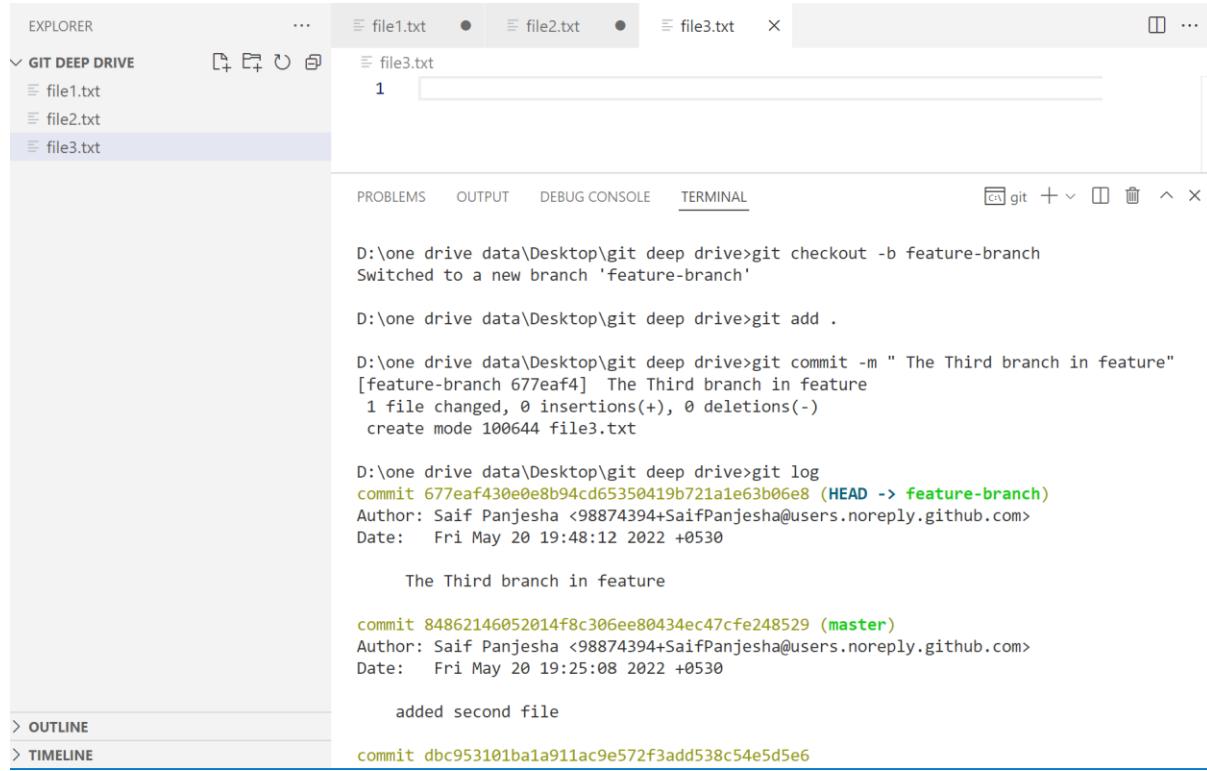
commit 7faede4ff4b3e58ccb6580c10094065c7458367b
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date: Fri May 20 18:33:36 2022 +0530

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

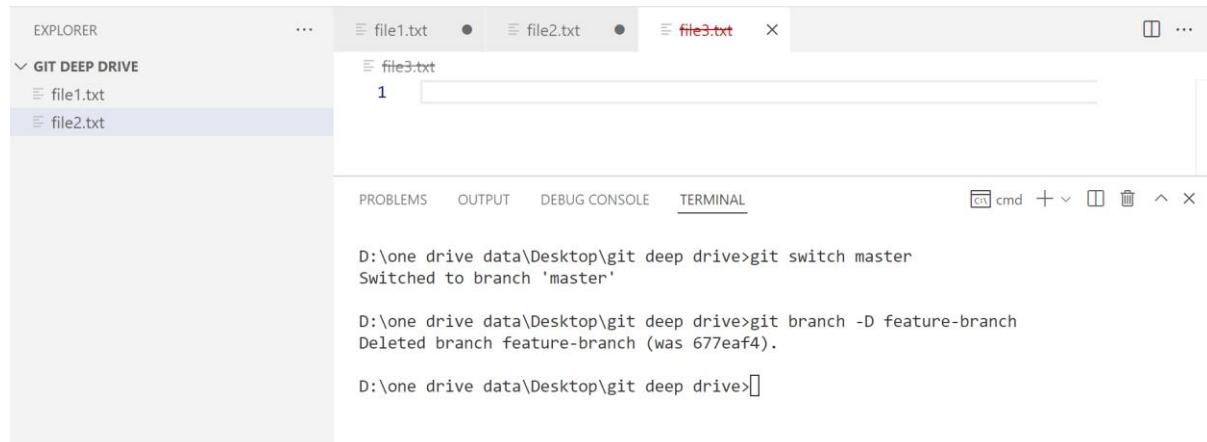


The screenshot shows the VS Code interface with the following details:

- EXPLORER** panel: Shows files file1.txt, file2.txt, and file3.txt.
- Git Deep Drive** panel: Shows the current branch is 'feature-branch'.
- Terminal** tab: Displays the command history:
 - 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 84862146052014f8c306ee80434ec47cfe248529 (master)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date: Fri May 20 19:25:08 2022 +0530
 - added second file
 - commit dbc953101ba1a911ac9e572f3add538c54e5d5e6
- OUTLINE** and **TIMELINE** buttons are visible at the bottom of the terminal panel.

Fig. created a new branch with file3.txt

Switch back to master branch:

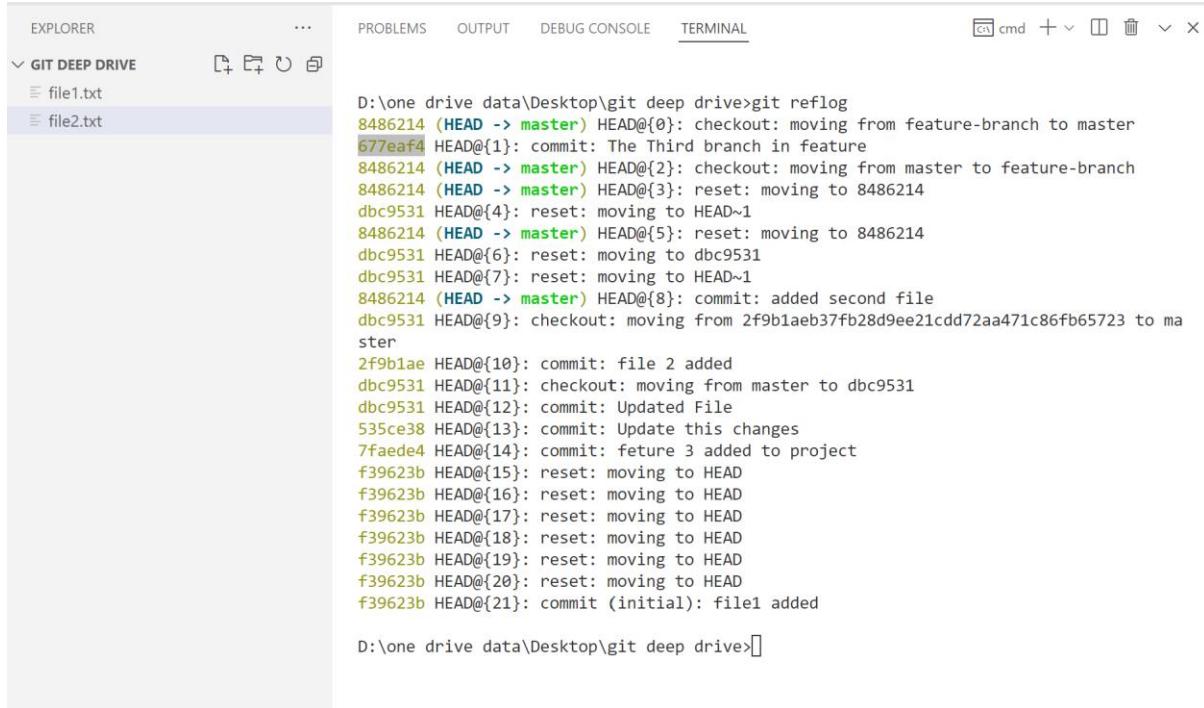


The screenshot shows the VS Code interface with the following details:

- EXPLORER** panel: Shows files file1.txt and file2.txt.
- Git Deep Drive** panel: Shows the current branch is 'master'.
- Terminal** tab: Displays the command history:
 - 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

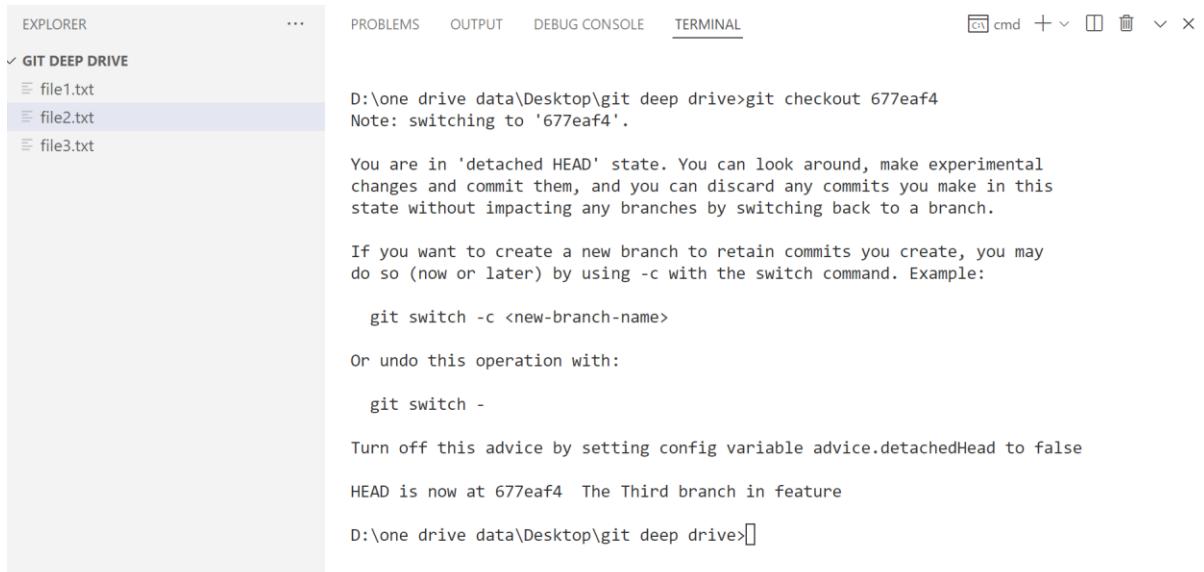
For restore the “**branches**” is differs from “**commit**”



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 2f9b1aeb37fb28d9ee21cddd72aa471c86fb65723 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

EXPLORER ... PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

GIT DEEP DRIVE

- file1.txt
- file2.txt
- file3.txt

```
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>git switch -c feature_branch
Switched to a new branch 'feature_branch'

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

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

> OUTLINE

Fig. successful added branches with “reflog”

EXPLORER ... PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

GIT DEEP DRIVE

- file1.txt
- file2.txt
- 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 84862146052014f8c306ee80434ec47cfe248529 (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
```

> OUTLINE

> TIMELINE

Fig. Commit history of current branch(feature_branch)

Combining Branches – What & Why?

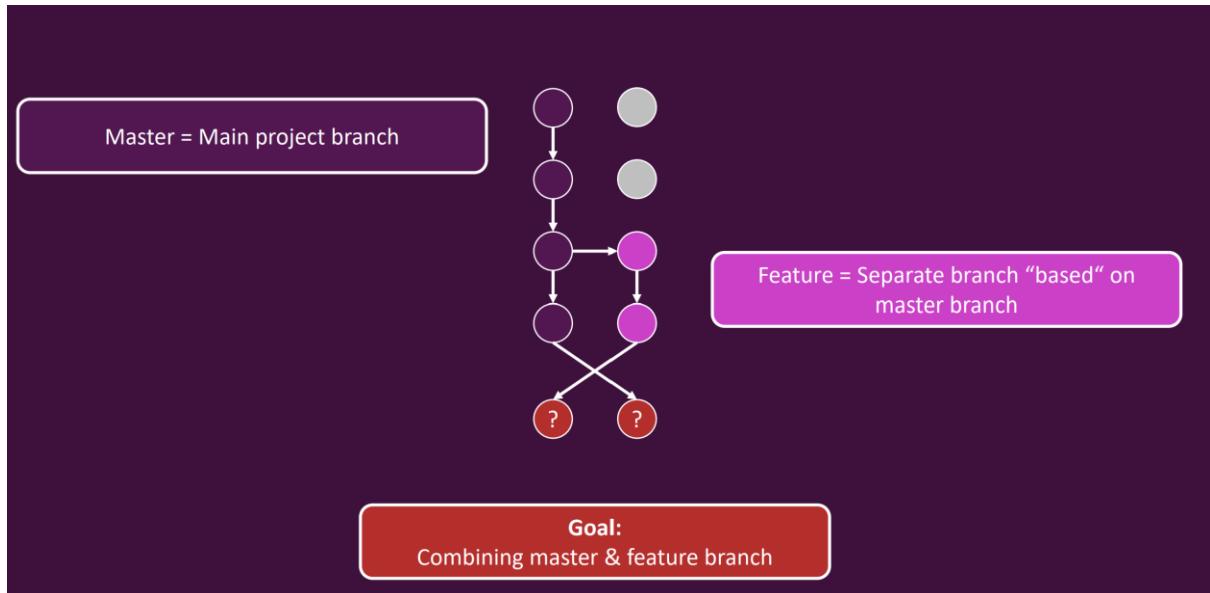


Fig. show Main Branch as Master and Separate Branch as feature

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.

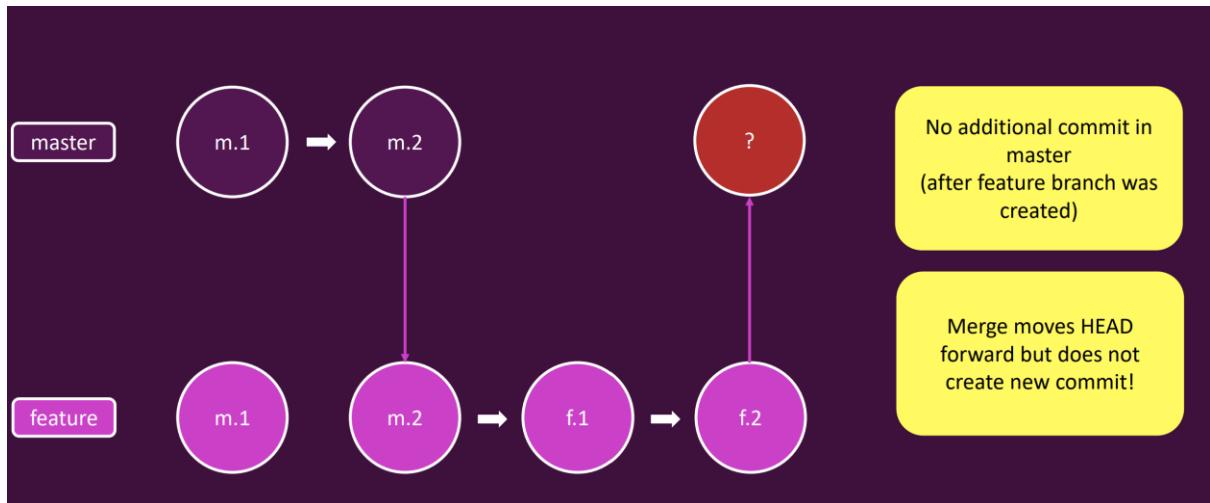


Fig. Shows Master & Feature -Merge("Fast-Forward")

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

A screenshot of a terminal window in a code editor (VS Code) showing the creation of a master branch. The terminal output is as follows:

```
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 branch named "master". Inside "master", there are two files: "m1.txt" and "m2.txt".
- BRANCHES**: Shows the current branch is "master".
- EDITOR**: Displays the contents of "m2.txt". The file contains the text "1".
- TERMINAL**: Shows 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
- COMMAND PALETTES**: Shows standard cmd palette icons.

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 branch named "feature". Inside "feature", there are two files: "f1.txt" and "f2.txt".
- OPEN EDITORS**: Shows the current editor is "f1.txt".
- TERMINAL**: Shows 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 89d84fcraf79eba8ce29224f1cf141fba98cefa34 (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
- COMMAND PALETTES**: Shows standard cmd palette icons.

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 git commit history:

```
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

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

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:

```
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:

A screenshot of the VS Code interface. The left sidebar shows an 'EXPLORER' view with a tree structure under 'OPEN EDITORS' and 'BRANCHES'. The 'feature' branch is selected. The right side shows the 'TERMINAL' tab open, displaying the command line output of a git merge operation. The output shows the command 'git merge feature' being run, followed by 'Updating f32b946..89d84fc' and 'Fast-forward' message. It lists file changes: 'feature/f1.txt | 0', 'feature/f2.txt | 0', '2 files changed, 0 insertions(+), 0 deletions(-)', 'create mode 100644 feature/f1.txt', and 'create mode 100644 feature/f2.txt'. The terminal ends with 'D:\one drive data\Desktop\Branches>'. The status bar at the bottom indicates the current branch is 'master'.

Fig. shows merged with master branch as fast - forward

A screenshot of the VS Code interface. The left sidebar shows an 'EXPLORER' view with a tree structure under 'OPEN EDITORS' and 'BRANCHES'. The 'feature' branch is selected. The right side shows the 'TERMINAL' tab open, displaying the command line output of a git log operation. The output shows a series of commits: a merge commit from 'feature' to 'master' (commit 89d84fc), followed by two commits from 'feature': 'f2 added' (commit b6fc683a2918b3f8b1da9198426b888192c4df43) and 'f1 added' (commit f32b9469d954ae7e4b2082b9f14435494697f390). Then, two commits from 'master': 'm2 added' (commit 09ac428c3a0b18850aec1c2f611898d0e42ab706) and 'm1 added' (commit 09ac428c3a0b18850aec1c2f611898d0e42ab706). The status bar at the bottom indicates the current branch is 'master'.

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 00becf01b922522dd53b7735859e92eb4746c31 (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 00becf01b9225222dd5b7735859e92eb4746c31 (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):

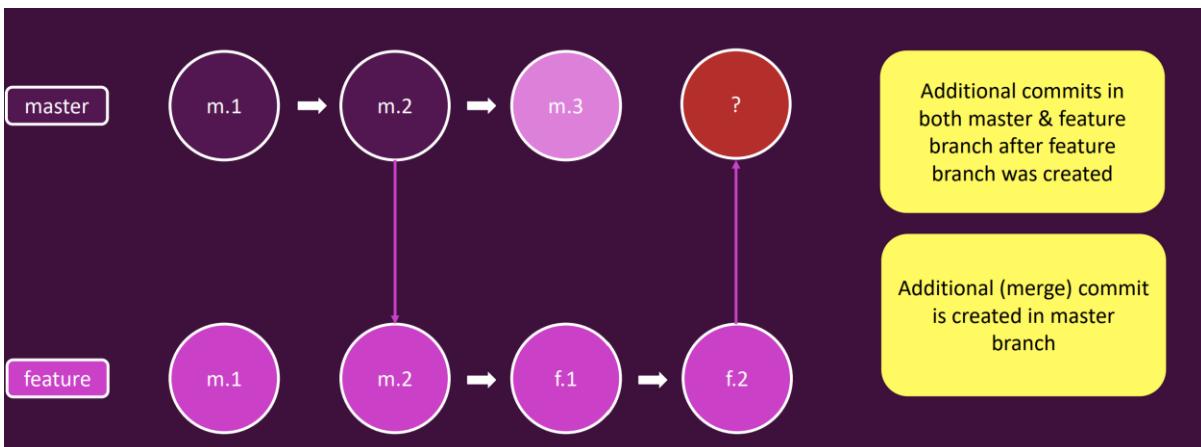


Fig. Shows Master & Feature – Recursive Merge(“Non -Fast-Forward”)

```

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**

```

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 fb7ff2718002060f6cd541f9c635b5ab6edbb26
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 fb7ff2718002060f6dc541f9c635b5ab6edbb26
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

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
cmd + ⌘ ⌄ ⌂ ⌅ ×

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

```

EXPLORER ... m1.txt m2.txt f1.txt f2.txt m3.txt X
OPEN EDITORS
BRANCHES
master > m3.txt
1

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
cmd + ⌘ ⌄ ⌂ ⌅ ×

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

```

EXPLORER ... m1.txt m2.txt f1.txt f2.txt m3.txt X ...
OPEN EDITORS
BRANCHES
master > m3.txt
1

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
cmd + ⌘ ⌄ ⌂ ⌅ ×

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 'git' tab selected in the top right. In the Explorer sidebar, under 'BRANCHES', the 'master' branch is expanded, showing three files: m1.txt, m2.txt, and m3.txt. The 'TERMINAL' tab displays the output of the command 'git log'. The log shows the following commits:

```

D:\one\drive\data\Desktop\Branches>git log
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 76e4b63e5cb83abb78c0f5e1861f2e1c11b4754a (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 terminal also shows the command 'git log' and the current directory 'D:\one\drive\data\Desktop\Branches>'.

Fig. full commit history shows in master branch

Let's do again –squash

The screenshot shows the VS Code interface with the 'cmd' tab selected in the top right. In the Explorer sidebar, under 'BRANCHES', the 'master' branch is expanded, showing three files: m1.txt, m2.txt, and m3.txt. The 'TERMINAL' tab displays the output of the command 'git reset --hard HEAD~1'. The log shows the following commits:

```

D:\one\drive\data\Desktop\Branches>git reset --hard HEAD~1
HEAD is now at 07e9d09 bc9a7726060ec9d2cc0685058c45d78ce (HEAD -> master)
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>[]

```

The terminal also shows the command 'git reset --hard HEAD~1' and the current directory 'D:\one\drive\data\Desktop\Branches>'.

Fig. switch to previous commit

The screenshot shows the VS Code interface with the Terminal tab selected. The Explorer sidebar on the left shows a 'BRANCHES' section with 'feature' and 'master' branches. The master branch contains files m1.txt, m2.txt, and m3.txt. 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 07e9d09bc9a7726060ec9d2cc06685058c45d78ce (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 Explorer sidebar on the left shows a 'BRANCHES' section with 'feature' and 'master' branches. The master branch contains files m1.txt, m2.txt, and m3.txt. 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

```

EXPLORER            ...
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL
cmd + v ⌂ v ×

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

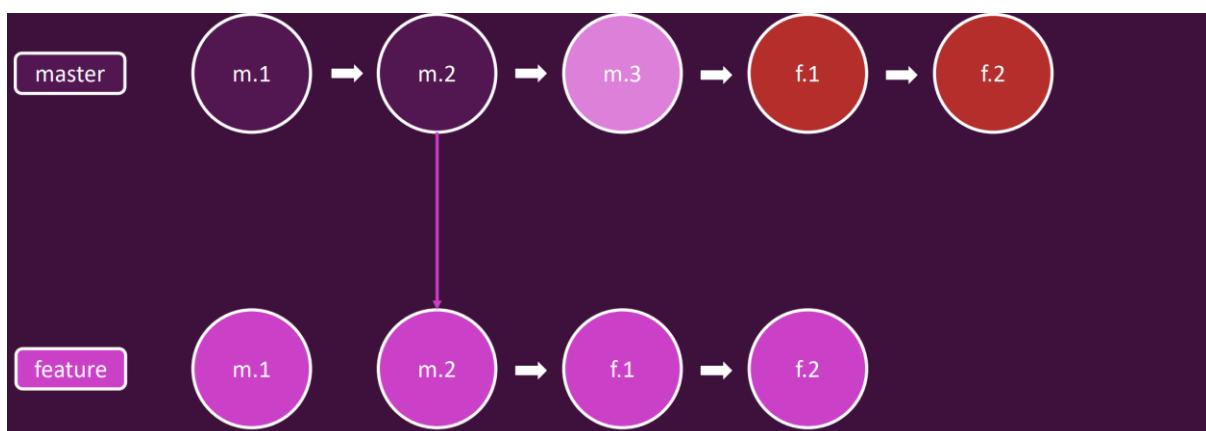


Fig. Shows Master & Feature of rebase

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.

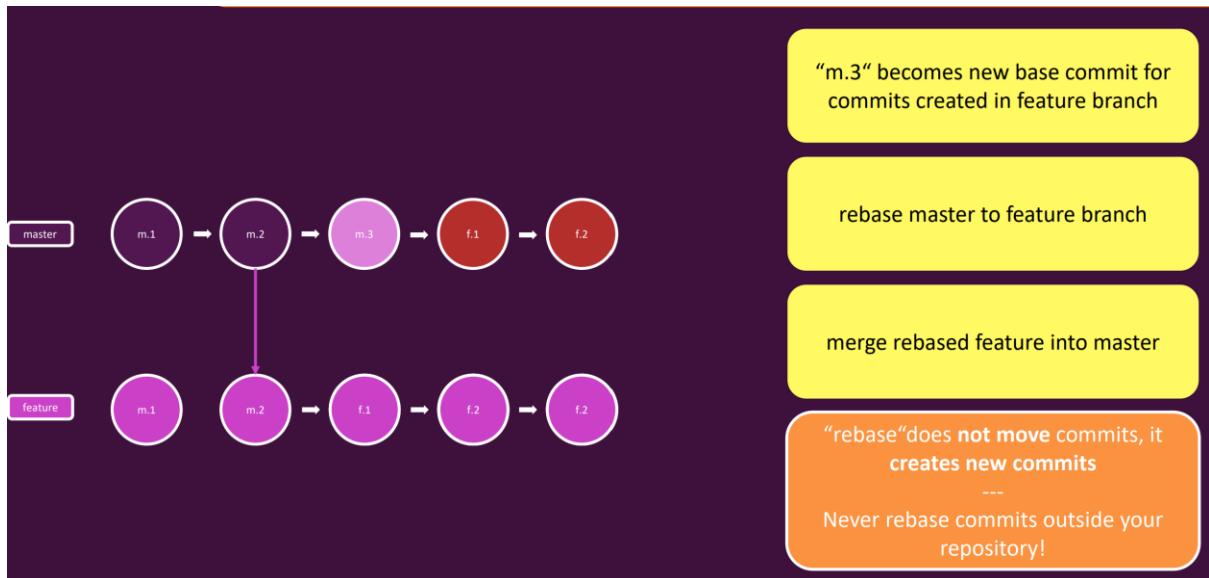


Fig. Shows using of rebase

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

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

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

```

D:\one drive data\Desktop\Branches>git log
commit 782e690a12faed175353a192e72b8fe1bd595de (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 5e29bffc5c3ffe06d05db7f7f1565e296b415380 (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 command and 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
```

Fig. fast-forward merging

When to apply rebase?

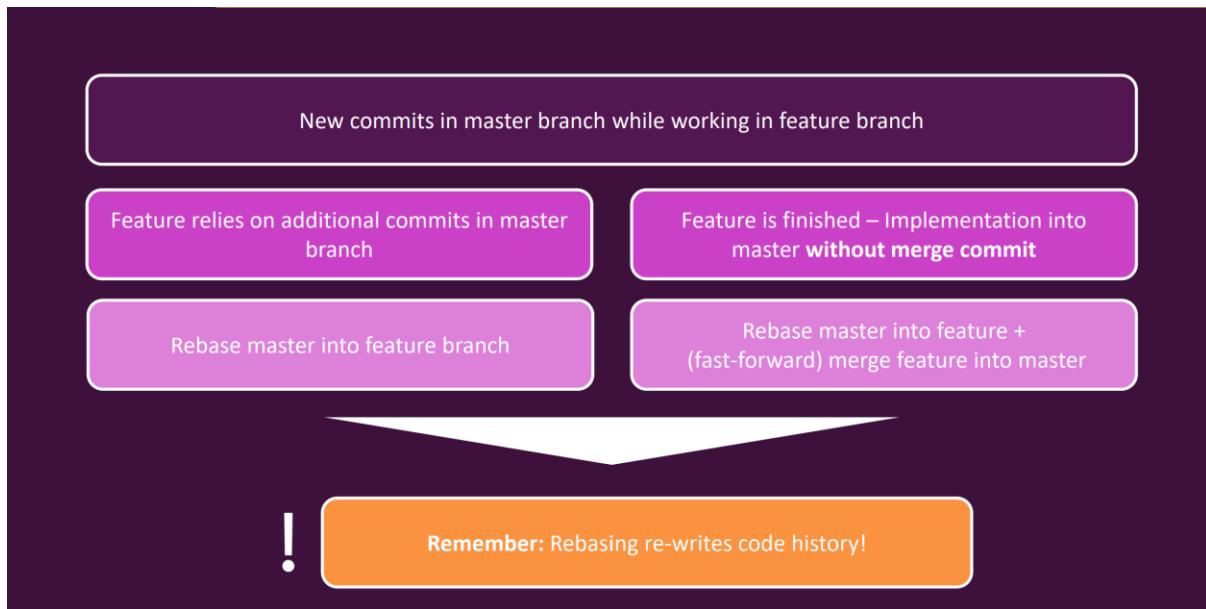


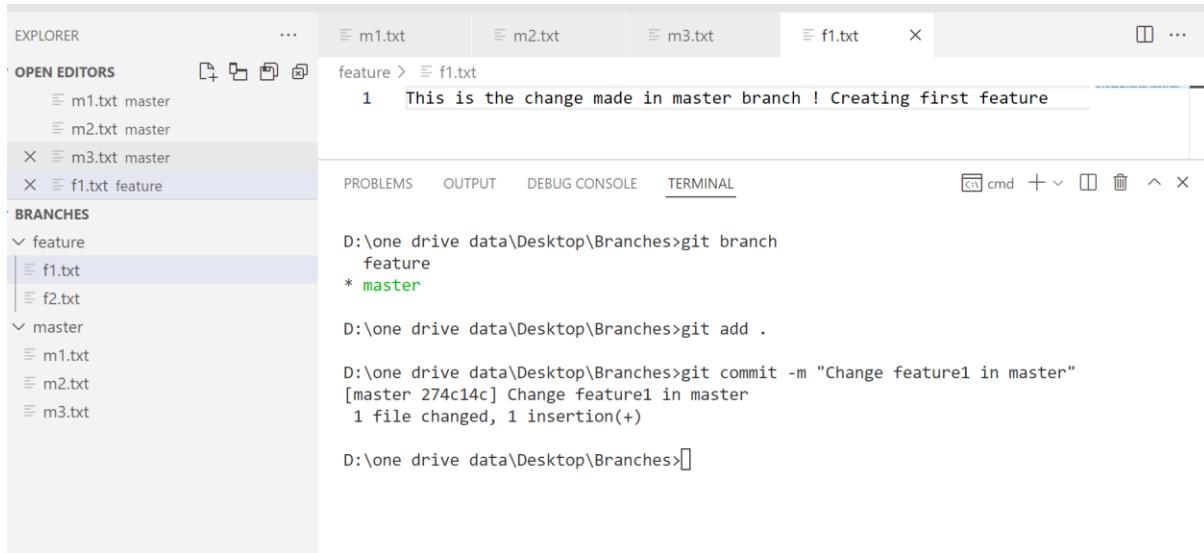
Fig. Workflow

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



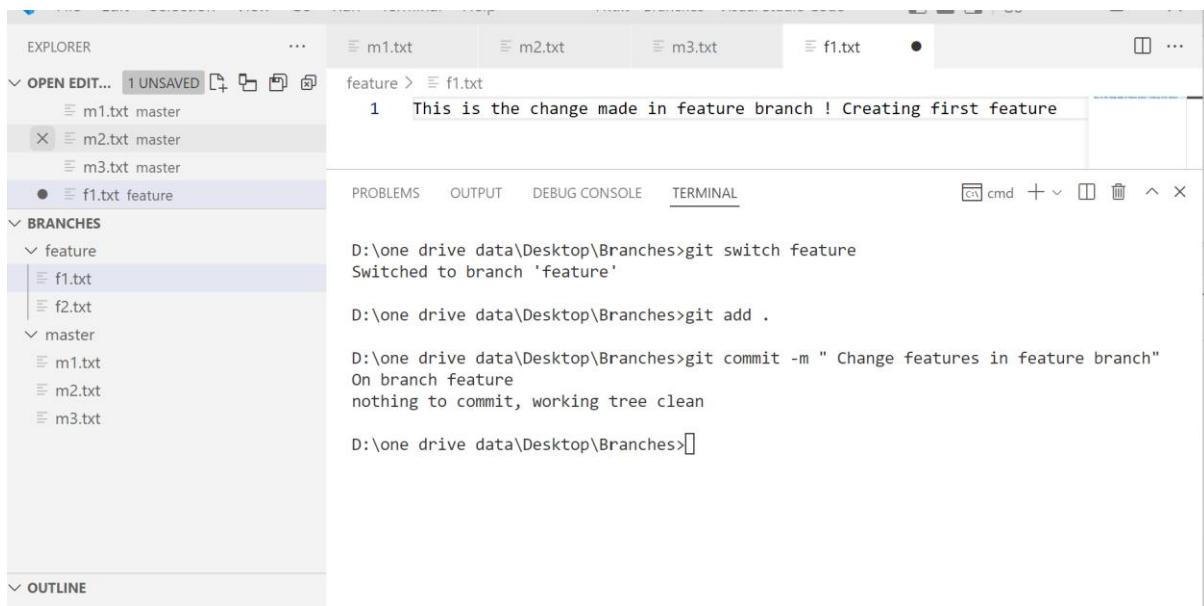
A screenshot of the Visual Studio Code interface. The left sidebar shows the 'OPEN EDITORS' list with 'f1.txt feature' selected. The 'BRANCHES' list shows 'feature' and 'master'. The 'feature' branch has files 'f1.txt' and 'f2.txt'. The 'master' branch has files 'm1.txt', 'm2.txt', and 'm3.txt'. The top right editor shows the content of 'f1.txt' with the following text:
1 This is the change made in master branch ! Creating first feature
The terminal below shows the command line history:
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



A screenshot of the Visual Studio Code interface. The left sidebar shows the 'OPEN EDIT...' list with 'f1.txt feature' selected. The 'BRANCHES' list shows 'feature' and 'master'. The 'feature' branch has files 'f1.txt' and 'f2.txt'. The 'master' branch has files 'm1.txt', 'm2.txt', and 'm3.txt'. The top right editor shows the content of 'f1.txt' with the following text:
1 This is the change made in feature branch ! Creating first feature
The terminal below shows the command line history:
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, which contains two conflicting changes:
 - Line 1: <<<< HEAD (Current Change)
 - Line 2: This is the change made in master branch ! Creating first feature
 - Line 3: =====
 - Line 4: This is the change made in feature branch ! Creating first feature
 - Line 5: >>>> feature (Incoming Change)
 - Line 6:
- BRANCHES**: Shows the current branch is feature.
- TERMINAL**: Shows the command history:


```
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.
```

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, identical to the previous screenshot.
- BRANCHES**: Shows the current branch is feature.
- TERMINAL**: Shows the command history and output of git status:


```
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")

Fig. git status shows how to handle 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 f1.txt with a conflict. 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)
      
```
- BRANCHES**: Shows the feature branch containing f1.txt.
- PROBLEMS**: Shows a conflict in f1.txt.
- OUTPUT**: Shows the command `git merge feature` and its output: "Auto-merging f1.txt" followed by a CONFLICT message.
- DEBUG CONSOLE**: Shows the command `git log --merge` and its output, including the commit hash 409837b77d3b174f5bd10a5d6672f26d4e6b21fd.
- TERMINAL**: Shows the command `git merge feature` being run.
- COMMANDS**: Shows standard terminal navigation commands like cmd, +, ^, x.

Fig. git log --merge 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 in the master branch, and f1.txt in the feature branch.
- OPEN EDITORS**: Displays f1.txt with a conflict. The content is identical to the previous screenshot.
- BRANCHES**: Shows the feature branch containing f1.txt.
- PROBLEMS**: Shows a conflict in f1.txt.
- OUTPUT**: Shows the command `git diff` and its output, including the diff output and the conflict markers - and +.
- DEBUG CONSOLE**: Shows the command `git diff` being run.
- TERMINAL**: Shows the command `git diff` being run.
- COMMANDS**: Shows standard terminal navigation commands like cmd, +, ^, x.

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 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 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 merge --abort

D:\one drive data\Desktop\Branches>

```

The terminal window is part of a larger interface with an Explorer sidebar on the left showing file structures for 'master' and 'feature' branches, and an 'OPEN EDITORS' section at the top.

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, showing f1.txt with the content "This is the change made in master branch ! Creating first feature".
- BRANCHES:** feature branch contains f1.txt and f2.txt; master branch contains m1.txt, m2.txt, m3.txt.
- PROBLEMS:** Shows git status output indicating unmerged paths and a merge commit message.
- OUTPUT:** Shows terminal output of git commands: status, add ., commit -m "merged feature and master in f1 file", and log.
- DEBUG CONSOLE:** Not visible in the screenshot.
- TERMINAL:** Shows the command line history from D:\one drive data\Desktop\Branches>git status to D:\one drive data\Desktop\Branches>git log.
- TIMELINE:** Shows a timeline of changes.

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, showing f1.txt with the content "This is the change made in master branch ! Creating first feature".
- BRANCHES:** feature branch contains f1.txt and f2.txt; master branch contains m1.txt, m2.txt, m3.txt.
- PROBLEMS:** Shows git log output listing commits for the master branch.
- OUTPUT:** Shows terminal output of git log command.
- DEBUG CONSOLE:** Not visible in the screenshot.
- TERMINAL:** Shows the command line history from D:\one drive data\Desktop\Branches>git log to D:\one drive data\Desktop\Branches>git log.
- TIMELINE:** Shows a timeline of changes.

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.

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

- EXPLORER:** Shows files `m1.txt`, `m2.txt`, `m3.txt`, and `f1.txt`. `m1.txt` is open in the editor.
- OPEN EDITORS:** Shows `m1.txt` (master), `m2.txt` (master), `m3.txt` (master), and `f1.txt` (feature).
- BRANCHES:** Shows branches `feature` and `master`. `m1.txt` is selected in the master branch.
- TERMINAL:** Shows the following command-line history:

```
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 "Working on m1"
[master bc75b55] Working on m1
  1 file changed, 1 insertion(+)

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

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

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

- EXPLORER**: Shows branches: feature (f1.txt, f2.txt, f3.txt, f4.txt), master (m1.txt, m2.txt, m3.txt), and f1.txt feature.
- OPEN EDITORS**: 1 UNSAVED (f4.txt)
- BRANCHES**: feature (f1.txt, f2.txt, f3.txt, f4.txt), master (m1.txt, m2.txt, m3.txt).
- OUTLINE**: No symbols found in document 'f4.txt'.
- TERMINAL**:


```
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**: Shows branches: master (m1.txt, f3.txt, f4.txt), feature (f1.txt, f2.txt, f3.txt, f4.txt), and f1.txt feature.
- OPEN EDITORS**: 1 UNSAVED (m1.txt)
- BRANCHES**: feature (f1.txt, f2.txt, f3.txt, f4.txt), master (m1.txt, m2.txt, m3.txt).
- OUTLINE**: No symbols found in document 'f4.txt'.
- TERMINAL**:


```
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 'TERMINAL' tab active. The terminal window displays the following git log output:

```
D:\one\drive\data\Desktop\Branches>git log
commit b7f6257adf5941db359f0d5dd99890533a1c40d0 (HEAD -> feature)
Author: Saif Panjesta <98874394+SaifPanjesta@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 Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date:   Sat May 21 19:46:02 2022 +0530

    Merge branch 'master' into feature

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

    f4 added

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

    f3 added

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

The Explorer sidebar shows the project structure with branches 'feature' and 'master'. 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. The Timeline sidebar shows a single commit for '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 'TERMINAL' tab active. The terminal window displays the following git commands and their output:

```
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>
```

The Explorer sidebar shows the project structure with branches 'feature' and 'master'. 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. The Timeline sidebar shows a single commit for 'Working on m1'.

Fig. specific commits has been added to 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

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

    Change feature1 in master

```

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
```

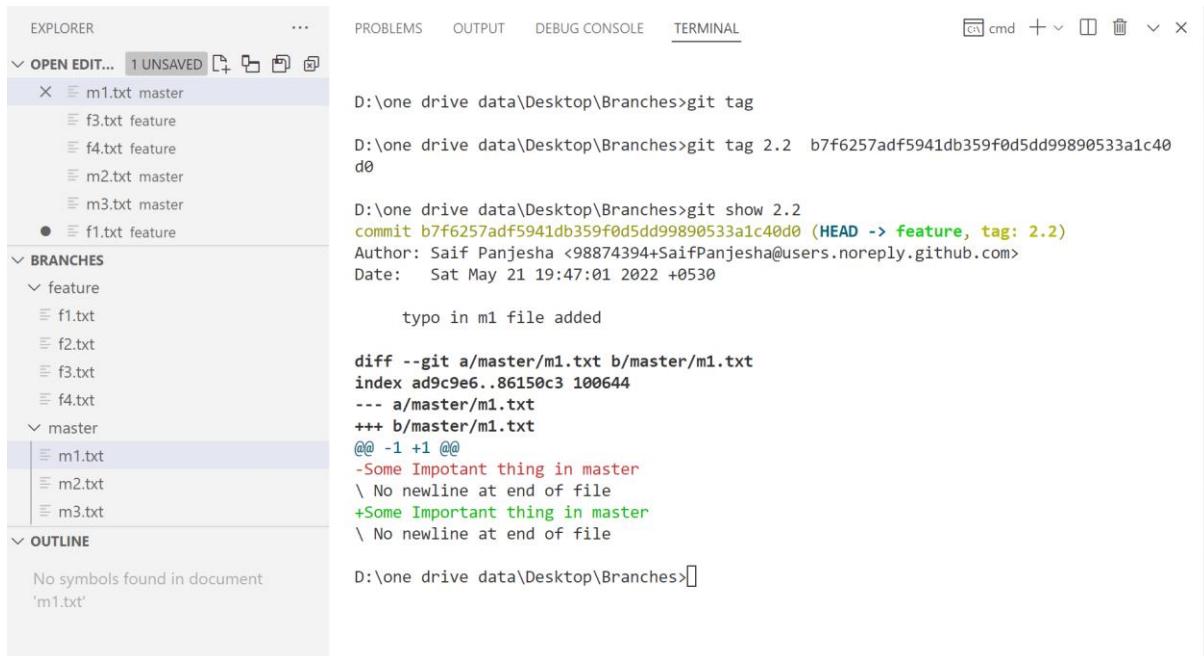
typo in m1 file added

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 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 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)

The screenshot shows the VS Code interface. On the left is the Explorer pane, which displays a file tree. It shows several files and folders: 'm1.txt' (master), 'f3.txt' (feature), 'f4.txt' (feature), 'm2.txt' (master), 'm3.txt' (master), and 'f1.txt' (feature). The 'f1.txt' entry is highlighted with a red dot. Below the file tree is a 'BRANCHES' section with a single entry 'feature'. On the right is the Terminal pane, which shows the command-line history:

```
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?

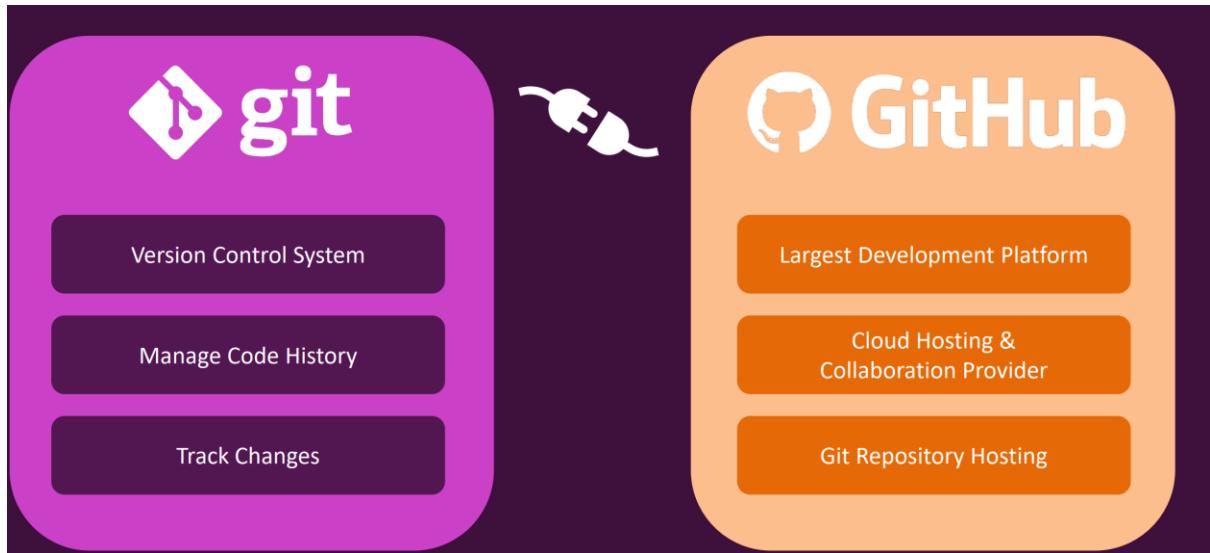


Fig. shows about Git and GitHub.

How Git and GitHub are connected?

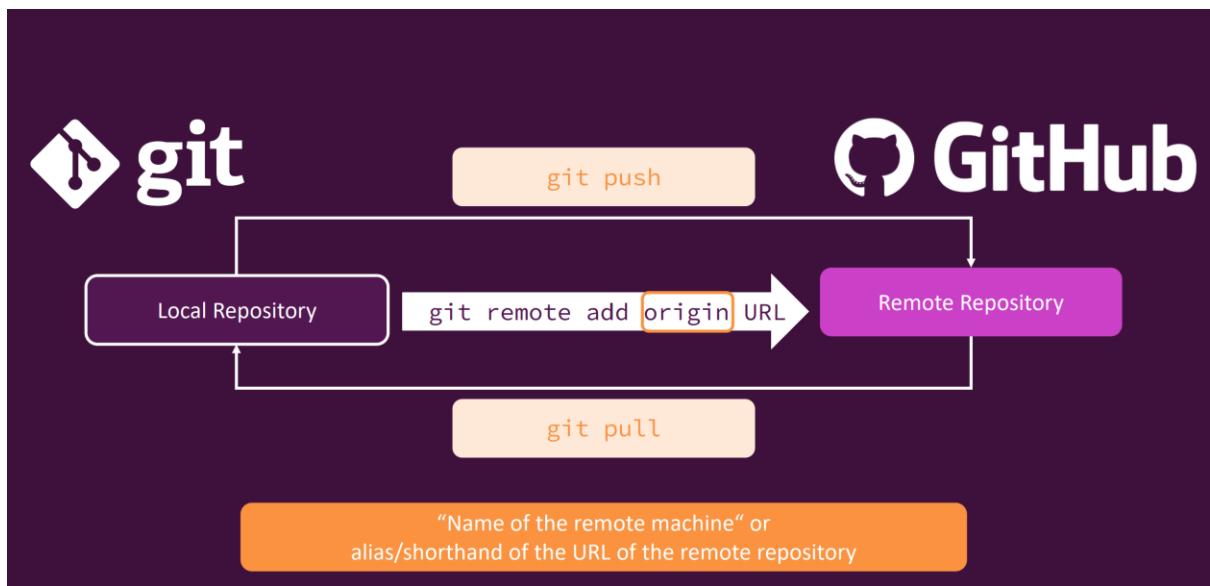


Fig. shows connecting git and GitHub - local to remote repository

Creating a GitHub account and introducing GitHub

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

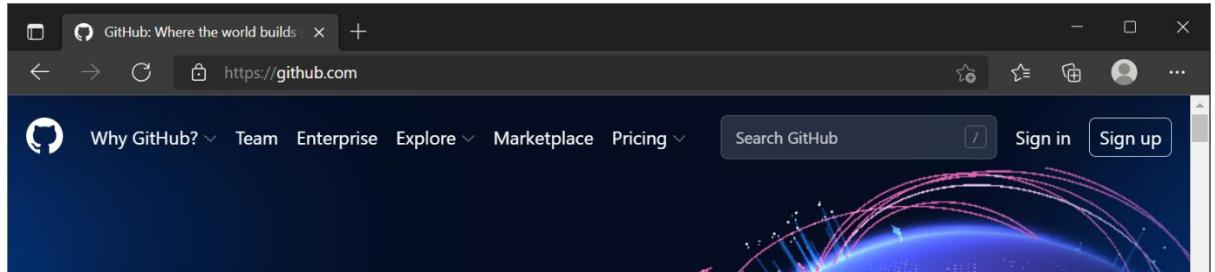


Fig. official site

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**.

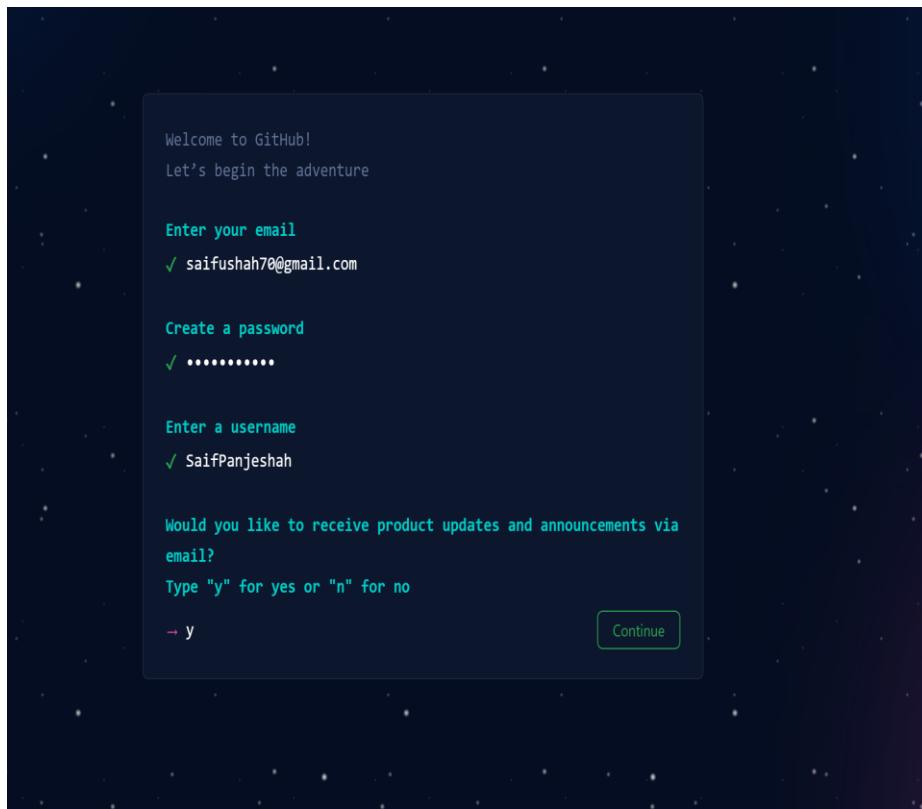


Fig. Creation of Username & Password

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

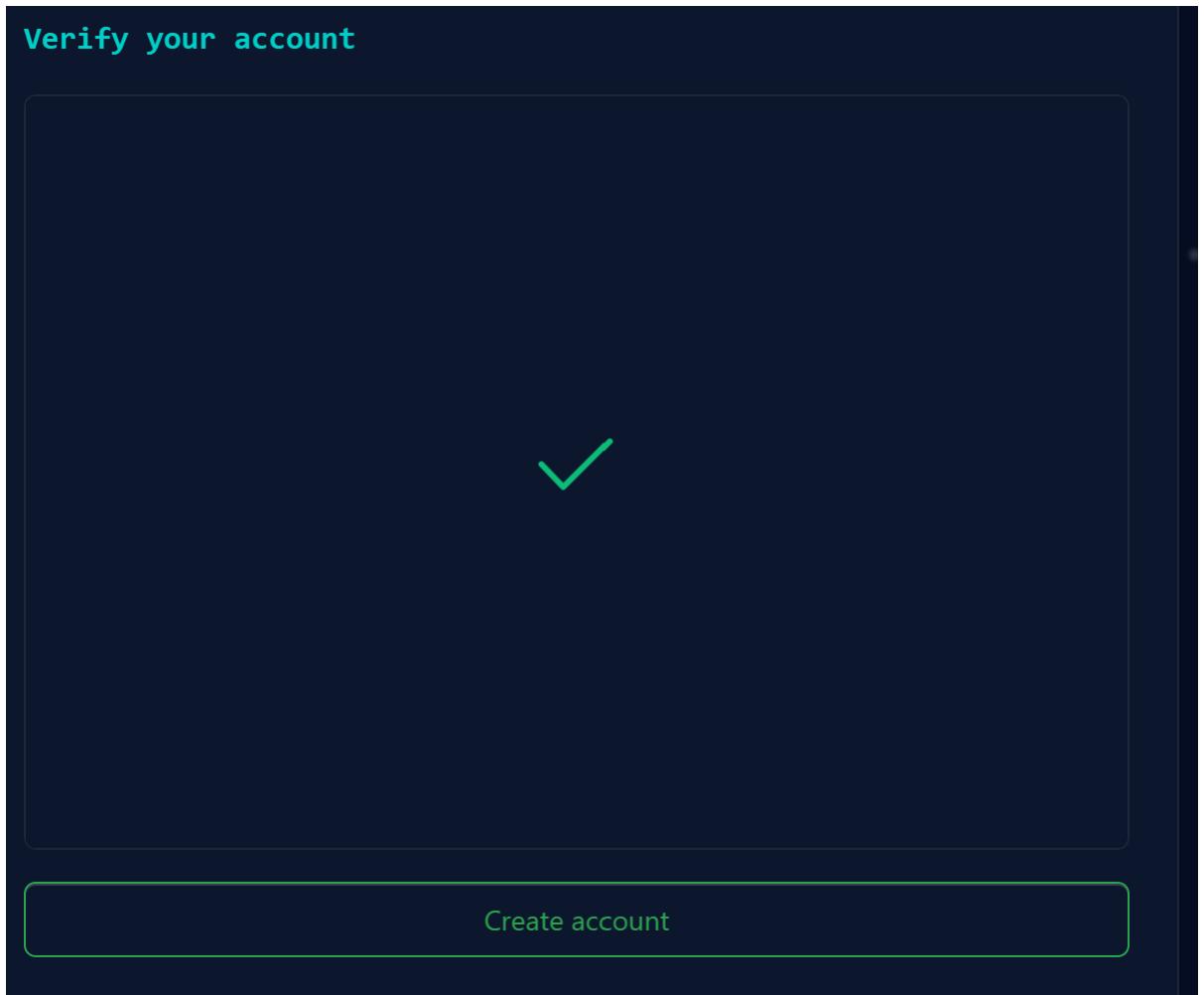


Fig. Account Verified

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**.

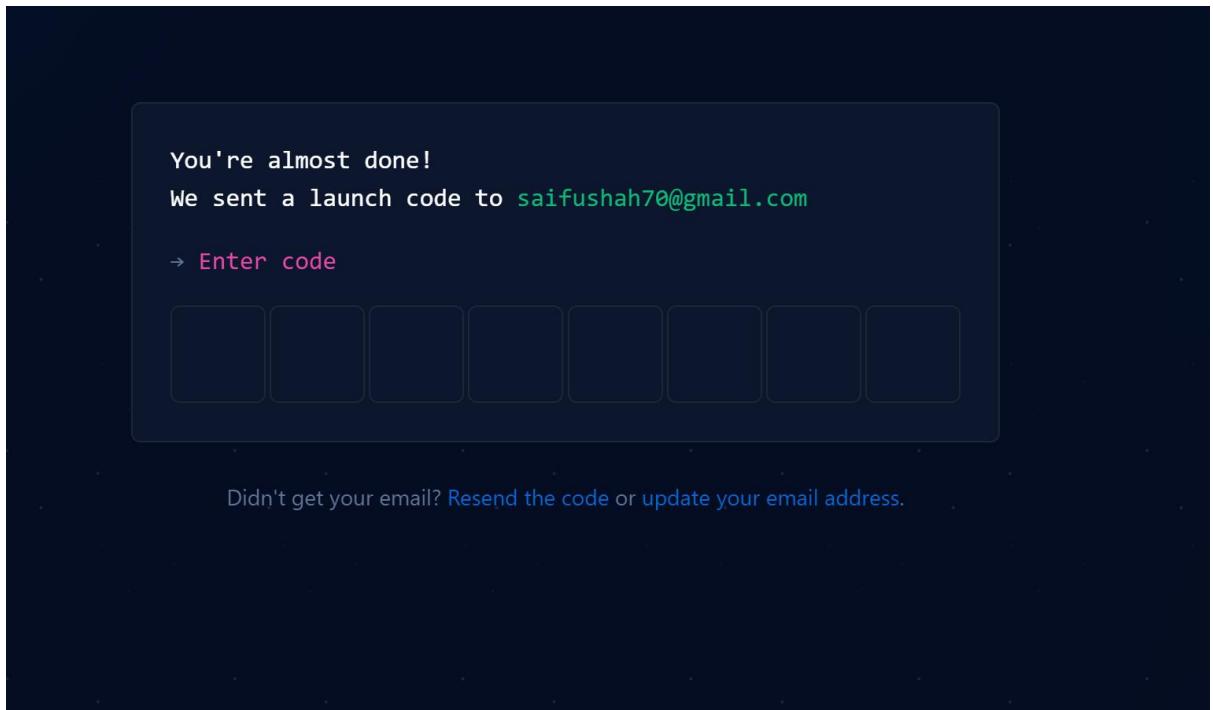


Fig. Enter the Code

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.

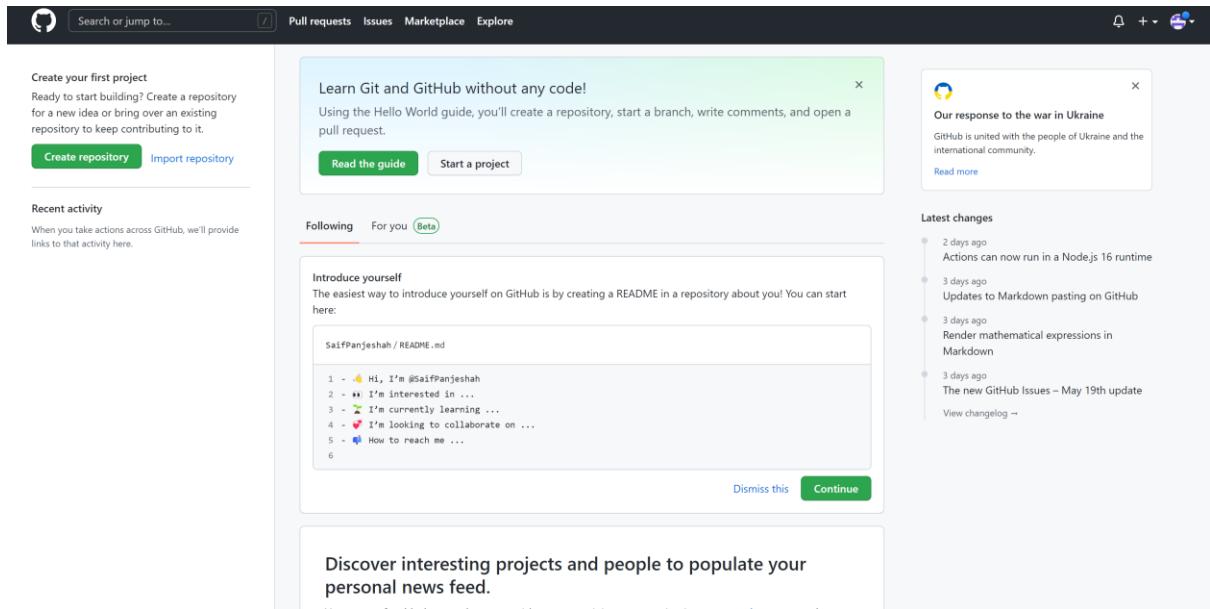


Fig. Account Creation Successful

Creating a repository:

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

The top screenshot shows the GitHub dashboard with a prominent 'Create repository' button. The bottom screenshot shows the user's profile page, which includes a large circular icon with a purple and white geometric pattern, and a message stating 'SaifPanjeshah doesn't have any public repositories yet.'

Fig. Creating repository

2. 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 .

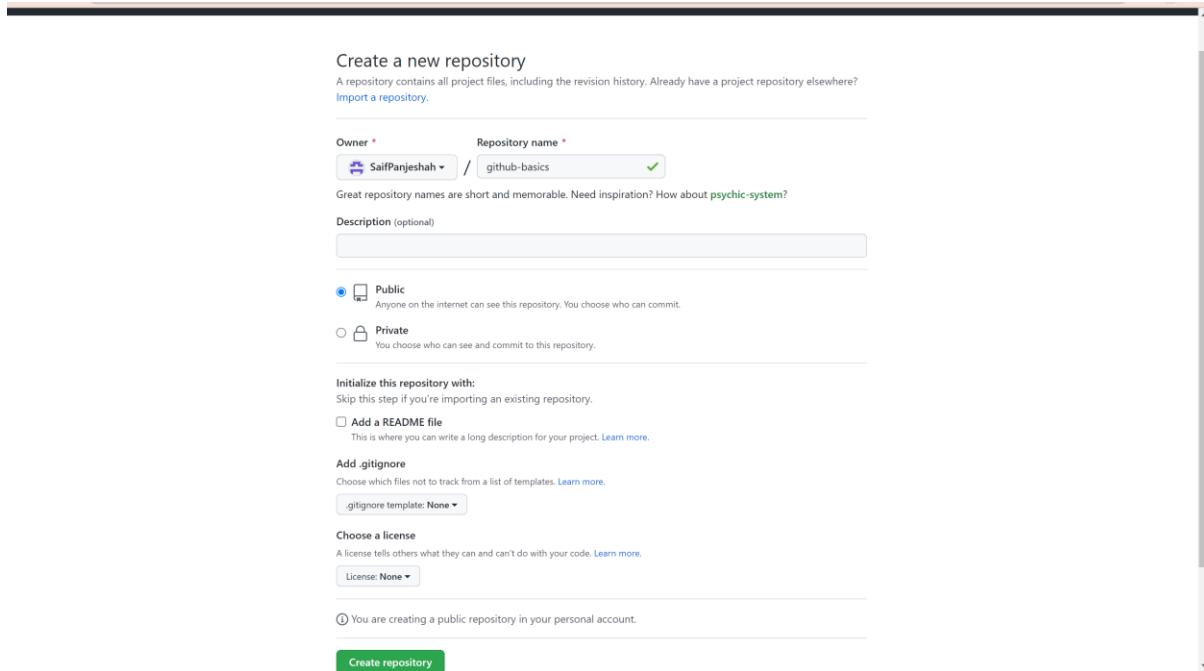


Fig. Created a new repository

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

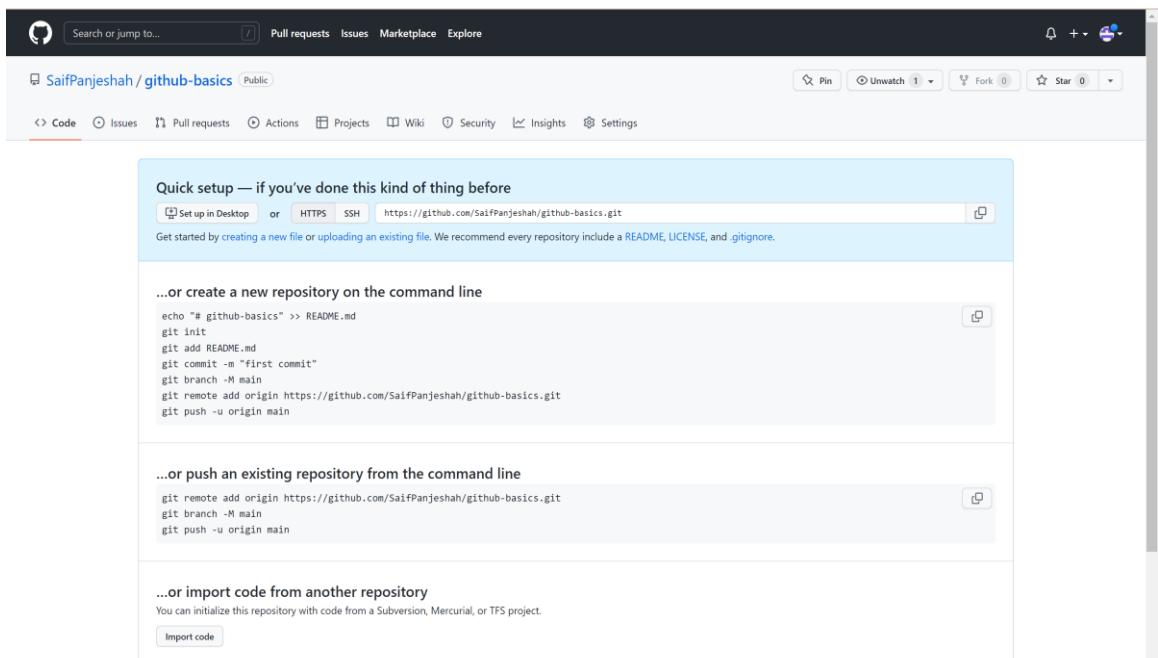


Fig. Setup git on GitHub

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

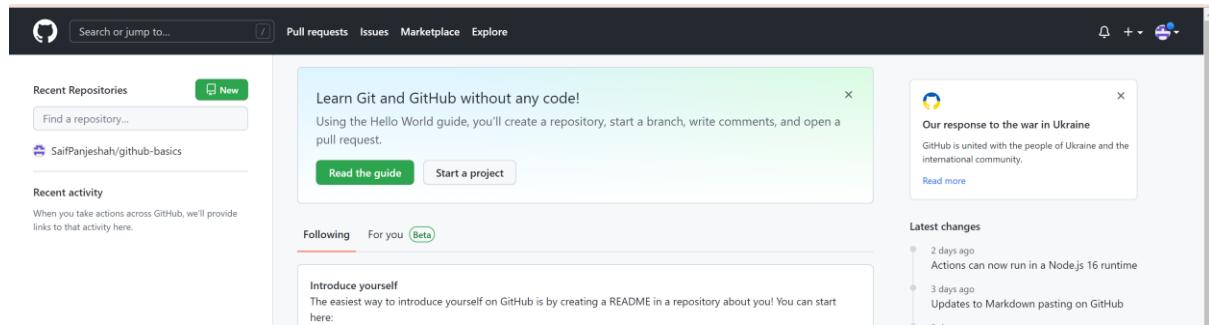


Fig. Repository Created

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

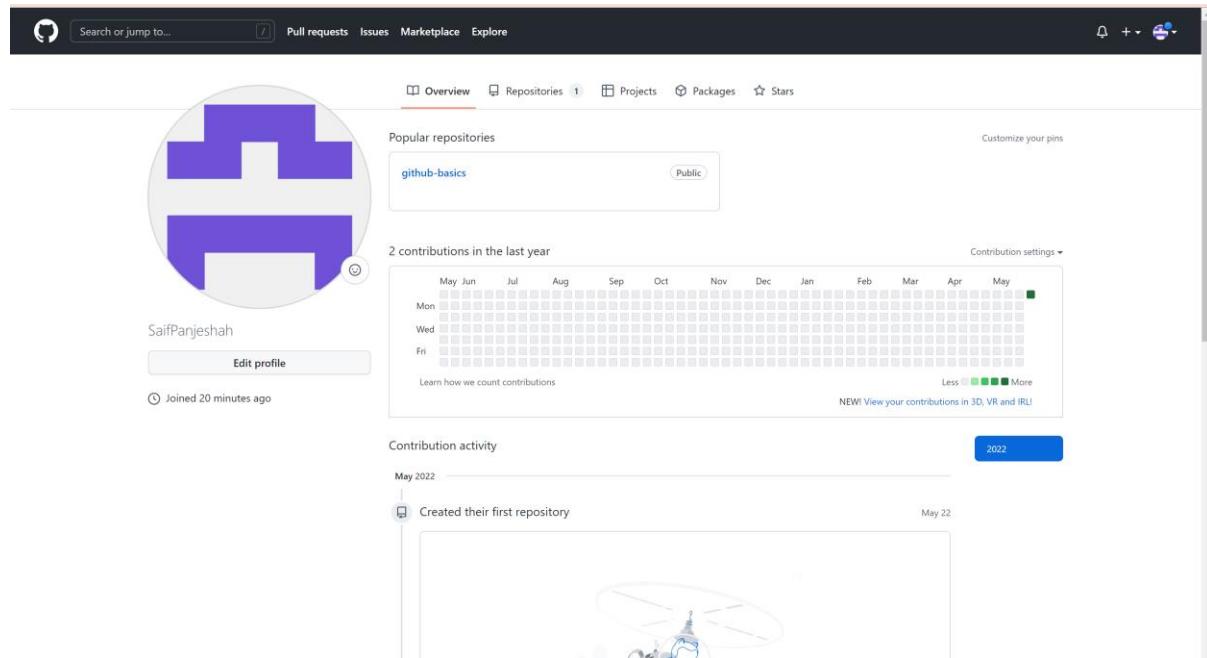


Fig. Successful remote repository added

Connecting Local and remote repository:

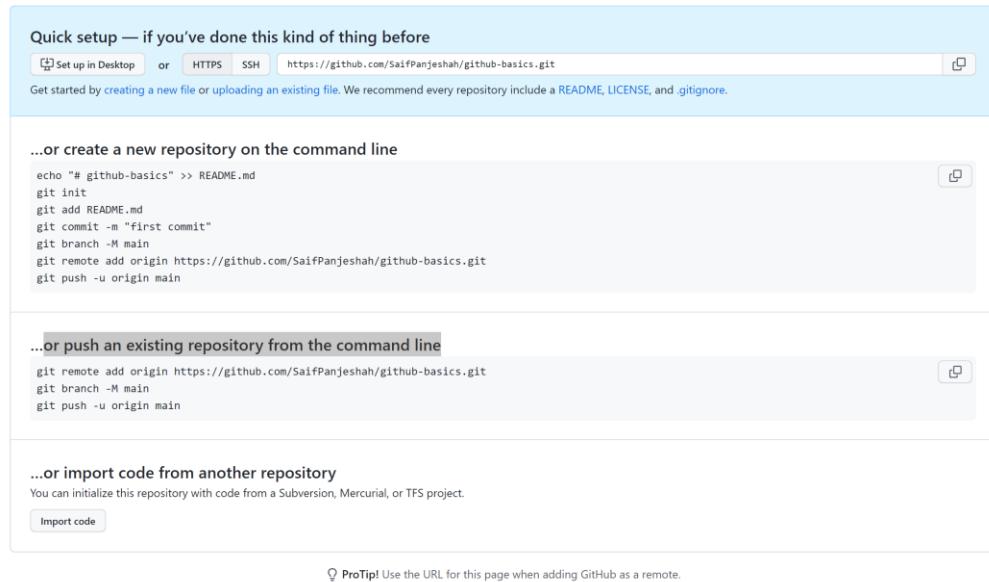


Fig. pushing repository from git

Git local repository

The screenshot shows the VS Code interface with the 'GITHUB...' extension installed. The Explorer sidebar shows a 'master' branch with a file named 'm1.txt'. The Terminal tab shows the following command history:

```
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
```

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

- EXPLORER**: Shows a folder named "GITHUB1-BASICS" containing a "master" branch and a file "m1.txt".
- TERMINAL**: Displays the command-line output of a git push operation.

```
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

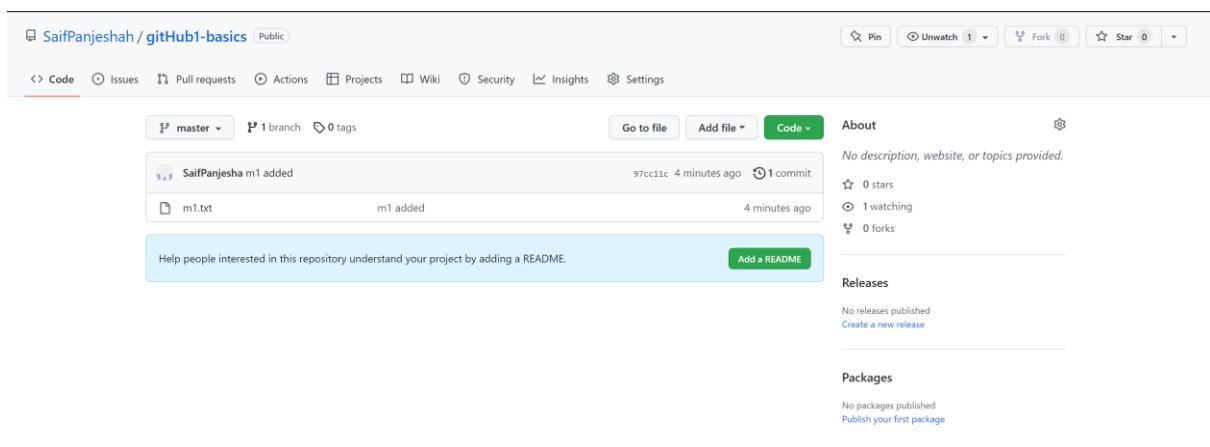


Fig. Successful push our local repository on GitHub via accessing through Internet browser.

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**.

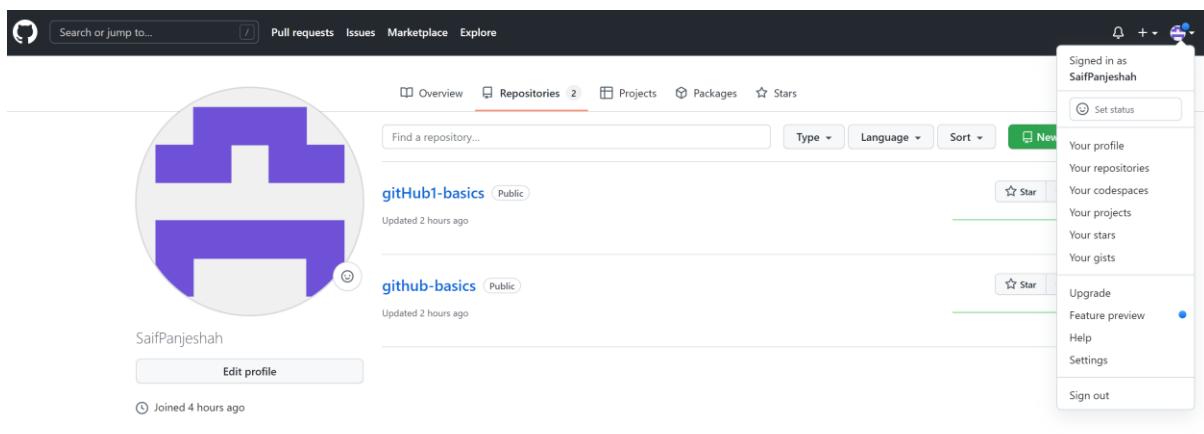


Fig. After verifying email address GitHub profile

- c. In the left sidebar, click **Developer settings**

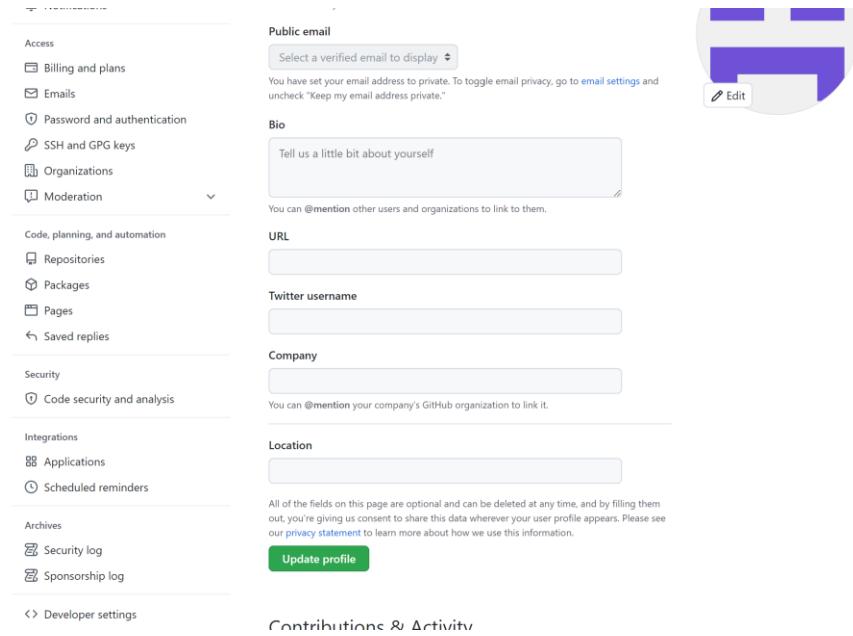


Fig. Setting to open developer setting

d. In the left sidebar, click **Personal access tokens**.

The screenshot shows the GitHub developer settings page at github.com/settings/tokens. The left sidebar has options for GitHub Apps, OAuth Apps, and Personal access tokens, with Personal access tokens selected. The main content area is titled "Personal access tokens" and contains a "Generate new token" button. Below it, there's a note about generating an API token for scripts or testing. At the bottom, there's a footer with links to 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 left sidebar has options for GitHub Apps, OAuth Apps, and Personal access tokens. The main content area is titled "New personal access token" and contains a note about personal access tokens. A "Note" field is filled with "Token For GitHub and Git Modules". Below it, there's a "What's this token for?" link.

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.

What's this token for?

Expiration *

90 days

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"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input checked="" type="checkbox"/> security_events	Read and write security events
<input checked="" type="checkbox"/> workflow	Update GitHub Action workflows
<input checked="" type="checkbox"/> write:packages	Upload packages to GitHub Package Registry
<input checked="" type="checkbox"/> read:packages	Download packages from GitHub Package Registry
<input checked="" type="checkbox"/> delete:packages	Delete packages from GitHub Package Registry
<input type="checkbox"/> admin:org	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"/> admin:public_key	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"/> admin:repo_hook	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"/> admin:org_hook	Full control of organization hooks
<input checked="" type="checkbox"/> gist	Create gists
<input checked="" type="checkbox"/> notifications	Access notifications
<input checked="" type="checkbox"/> user	Update ALL user data
<input checked="" type="checkbox"/> read:user	Read ALL user profile data

Fig. Permission for tokens excepts admin

h. Click **Generate token**.

<input checked="" type="checkbox"/> notifications	Access notifications
<input checked="" type="checkbox"/> user	Update ALL user data
<input checked="" type="checkbox"/> read:user	Read ALL user profile data
<input checked="" type="checkbox"/> user:email	Access user email addresses (read-only)
<input checked="" 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 checked="" 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

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.

Make sure to copy your personal access token now. You won't be able to see it again!

✓ ghp_wd0vty043COHAK71UIka0nDZLU80jB0T6k1N [Copy](#) [Delete](#)

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. Token Successful Generated

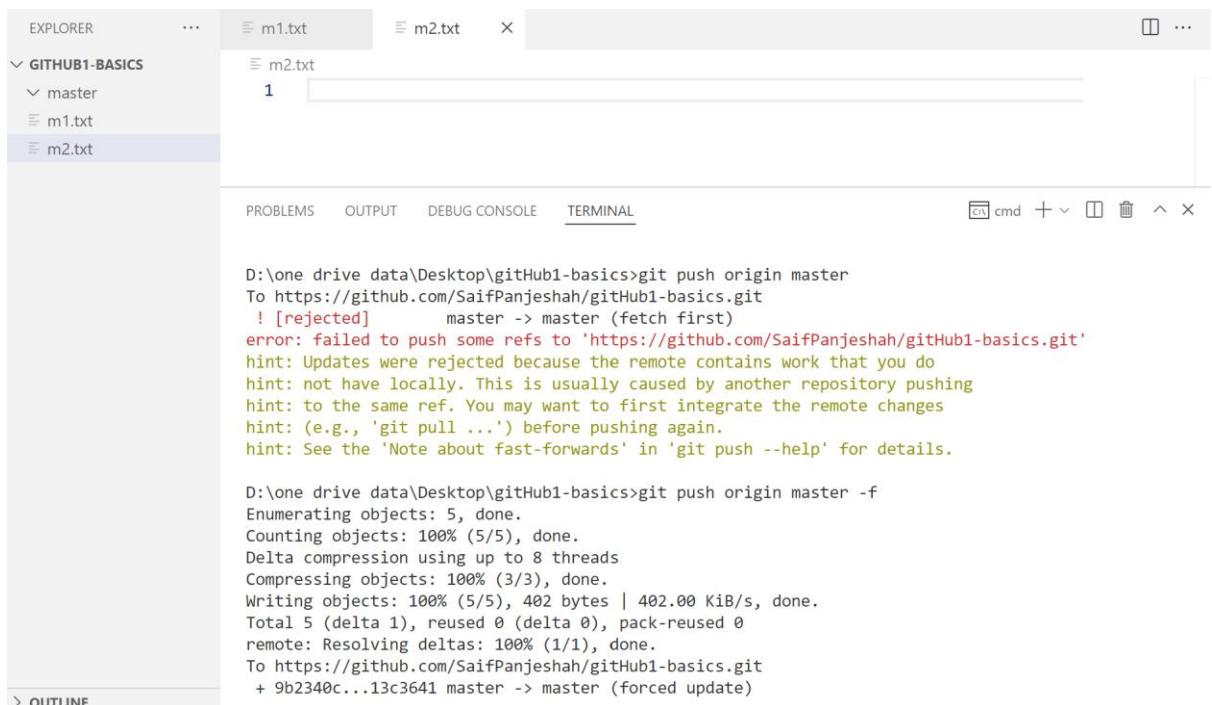
Pushing a Second Commit



The screenshot shows the VS Code interface. In the Explorer sidebar, there is a folder named "GITHUB..." containing a "master" branch with files "m1.txt" and "m2.txt". The "m2.txt" file is open in the editor, showing the number "1". Below the editor is a terminal window with the following command history:

```
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



The screenshot shows the VS Code interface. In the Explorer sidebar, there is a folder named "GITHUB1-BASICS" containing a "master" branch with files "m1.txt" and "m2.txt". The "m2.txt" file is open in the editor, showing the number "1". Below the editor is a terminal window with the following command history:

```
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

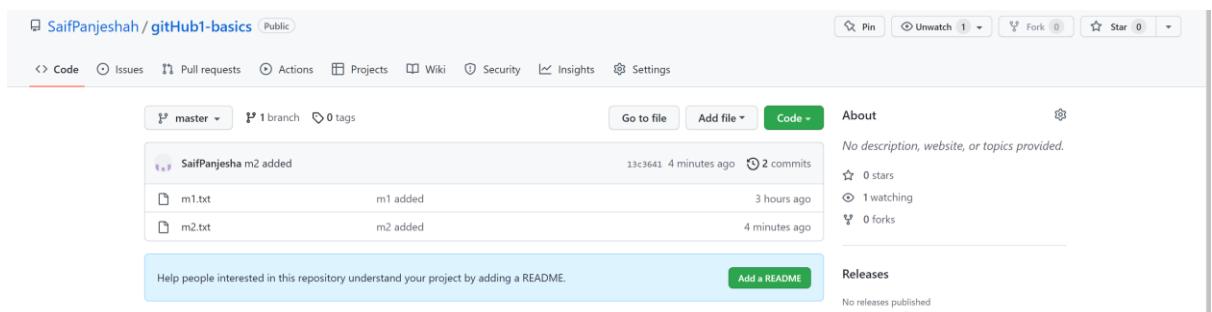
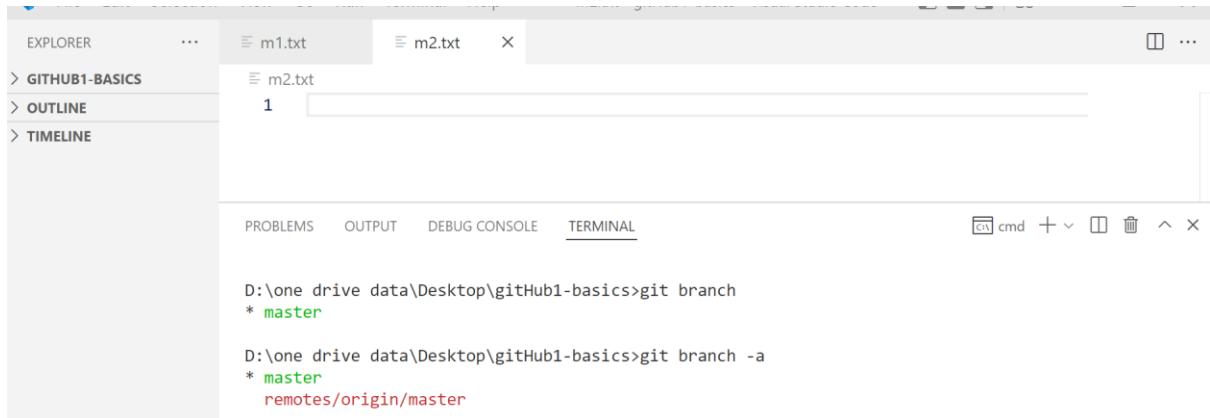


Fig. Successful on GitHub

From Local to remote Understanding the Workflow



A screenshot of a terminal window within a code editor interface. The terminal shows the command `git branch` being run twice. The first run shows a local `master` branch, and the second run shows a remote tracking branch `remotes/origin/master`. The terminal tabs at the bottom include PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL.

```
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

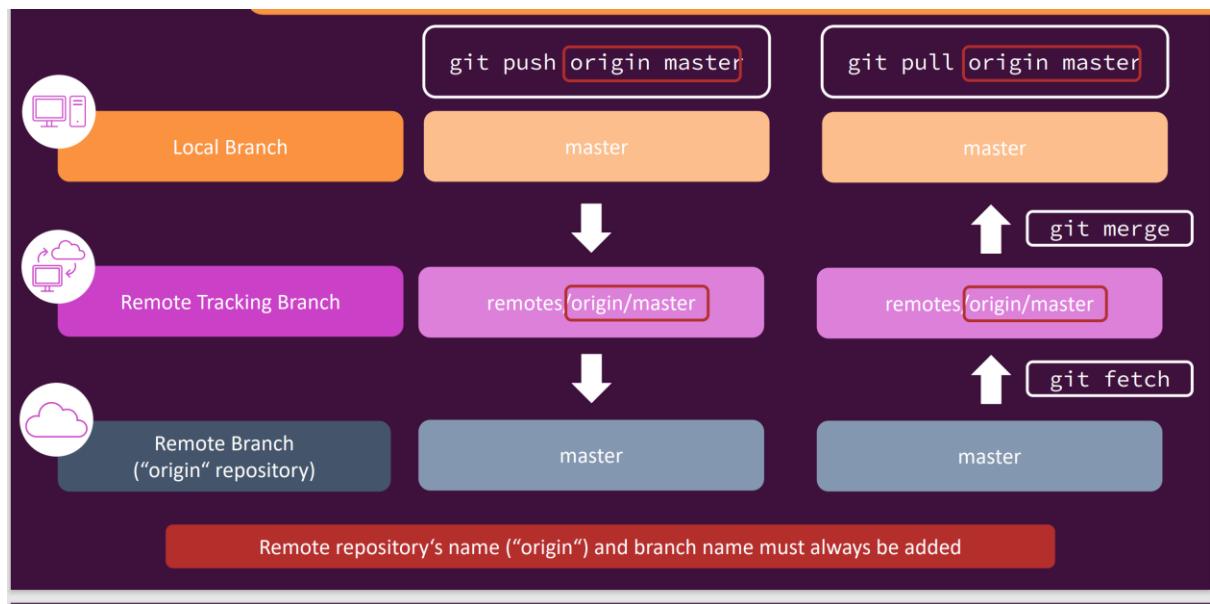
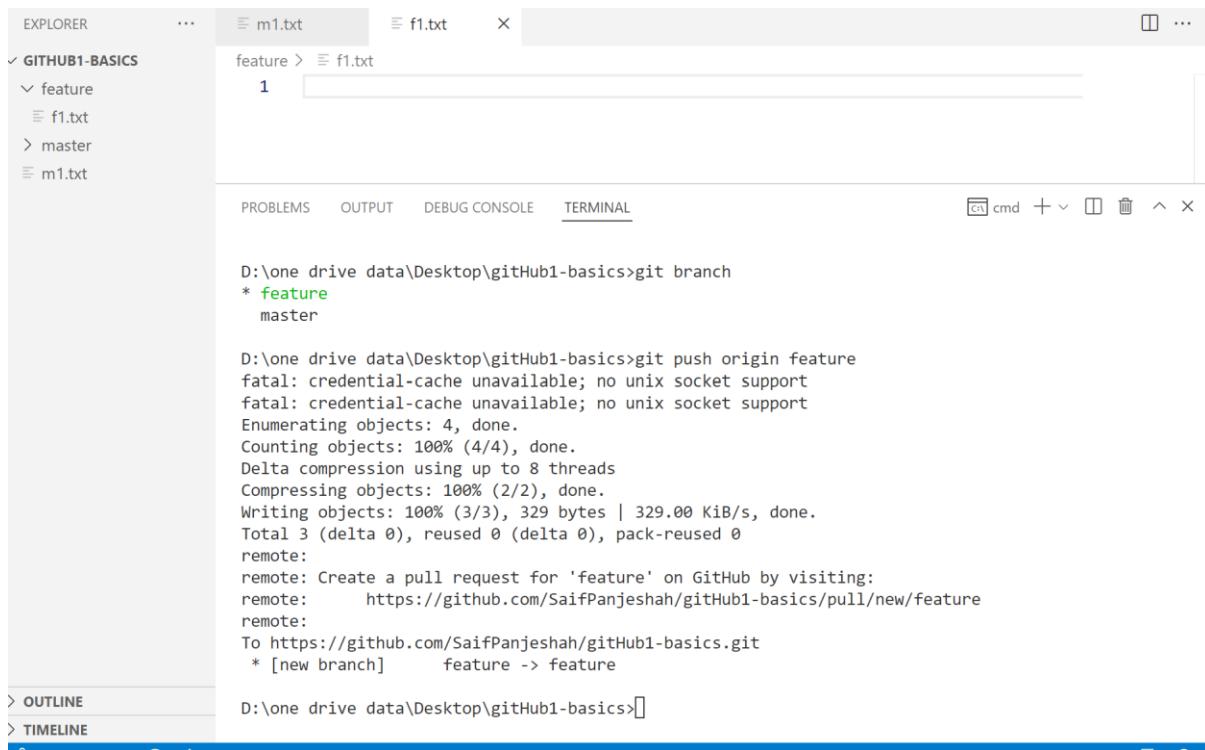


Fig. Local to remote Workflow

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 are two tabs: 'm1.txt' and 'f1.txt'. The 'f1.txt' tab is active, displaying the content '1'. Below the workspace are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', and 'TERMINAL'. The 'TERMINAL' tab is selected and shows the following command-line output:

```
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

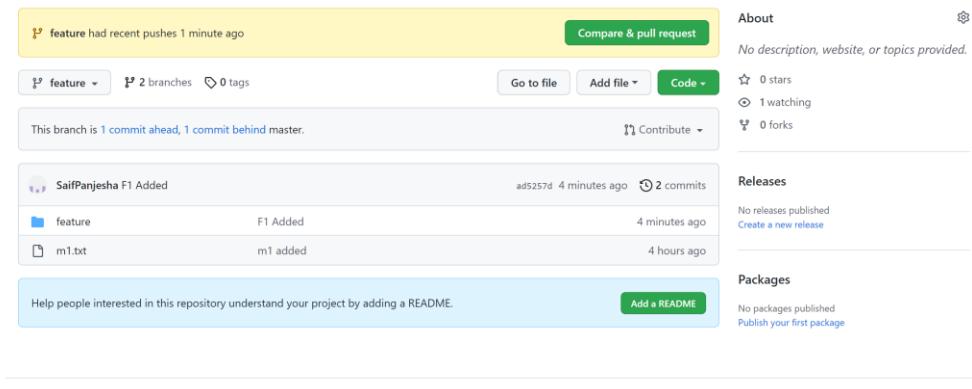


Fig. Successful added on GitHub

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

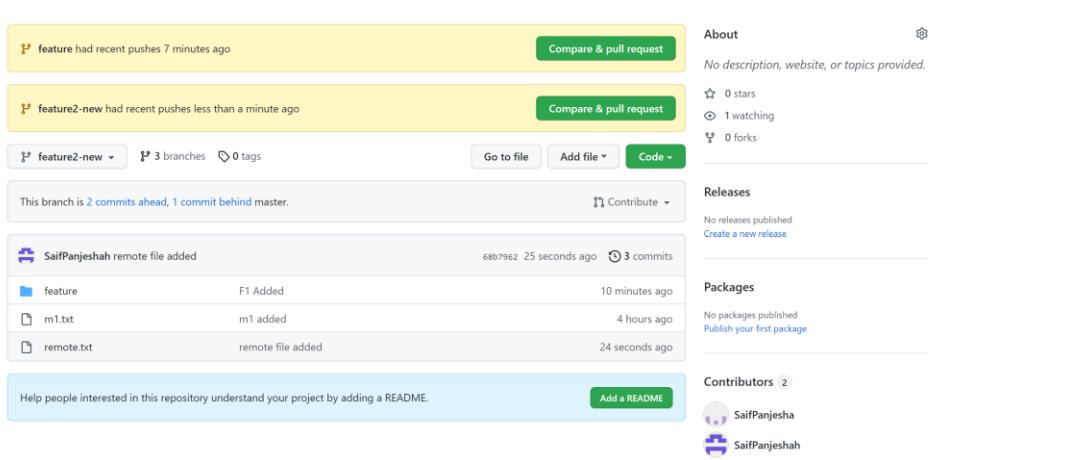


Fig. Created new Branch on Git Hub

The screenshot shows the VS Code interface with the 'TERMINAL' tab selected. The terminal window displays the following git commands and their output:

```

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-r
  ls-remote
  master
  remotes/origin/feature
  remotes/origin/master

D:\one drive data\Desktop\gitHub1-basics>

```

The terminal shows the addition of a new remote branch 'feature2-new' to the local repository.

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 'TERMINAL' tab selected. The terminal window displays the following git command and its output:

```

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

```

The terminal shows the execution of the 'git fetch origin' command, which retrieves the latest state from the remote repository.

Fig. Shows git fetch remote retrieves latest state

```

EXPLORER      PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL
GITHUB1-BASICS
> feature
> master
≡ m1.txt
≡ remote.txt

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

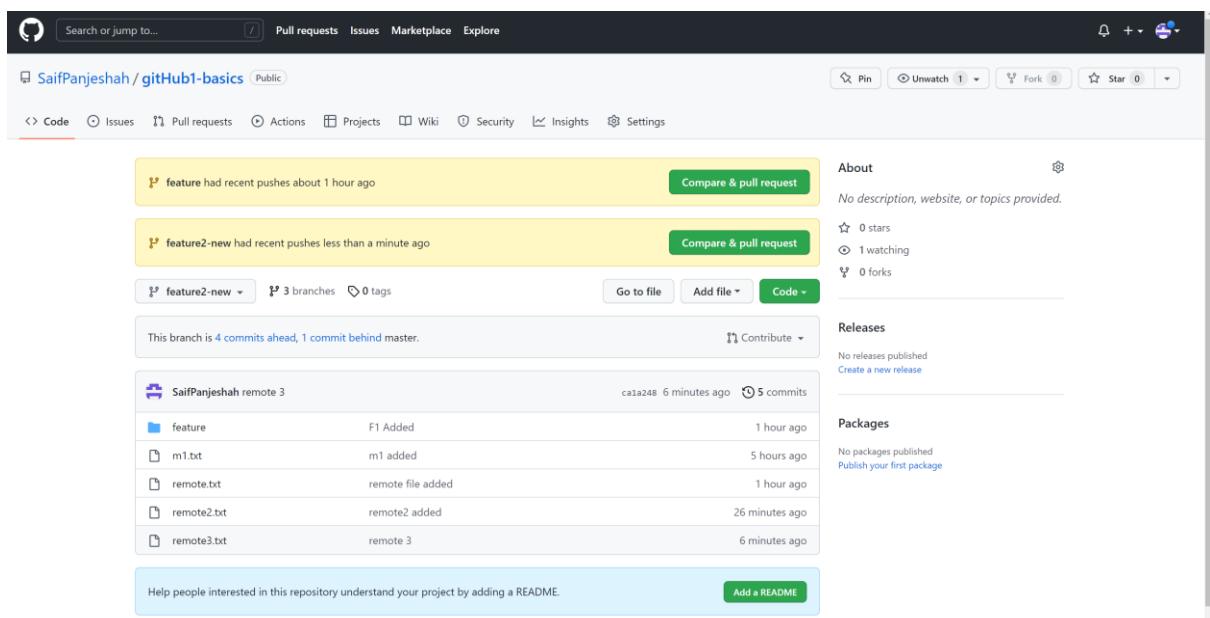
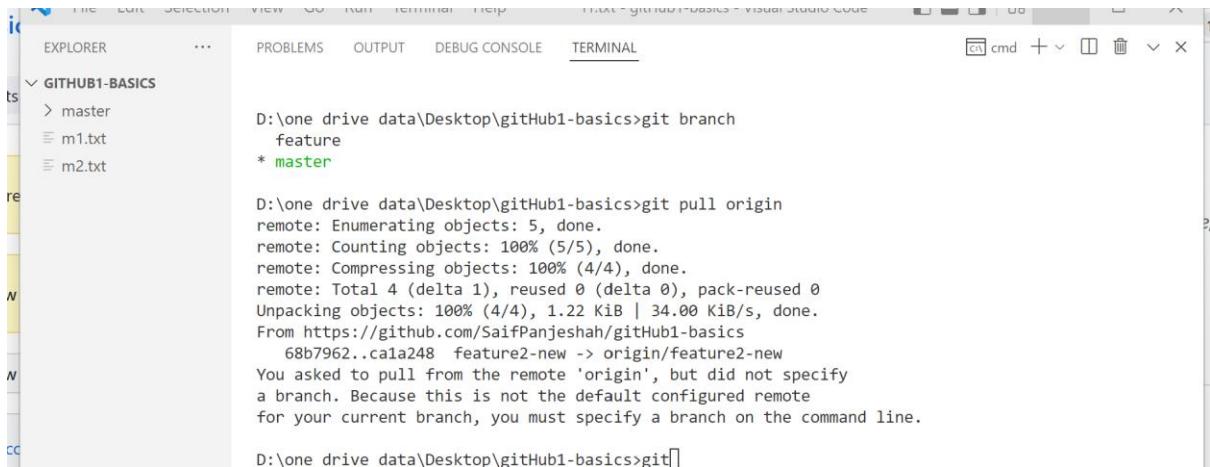


Fig. Shows Feature2 branch in GitHub

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)



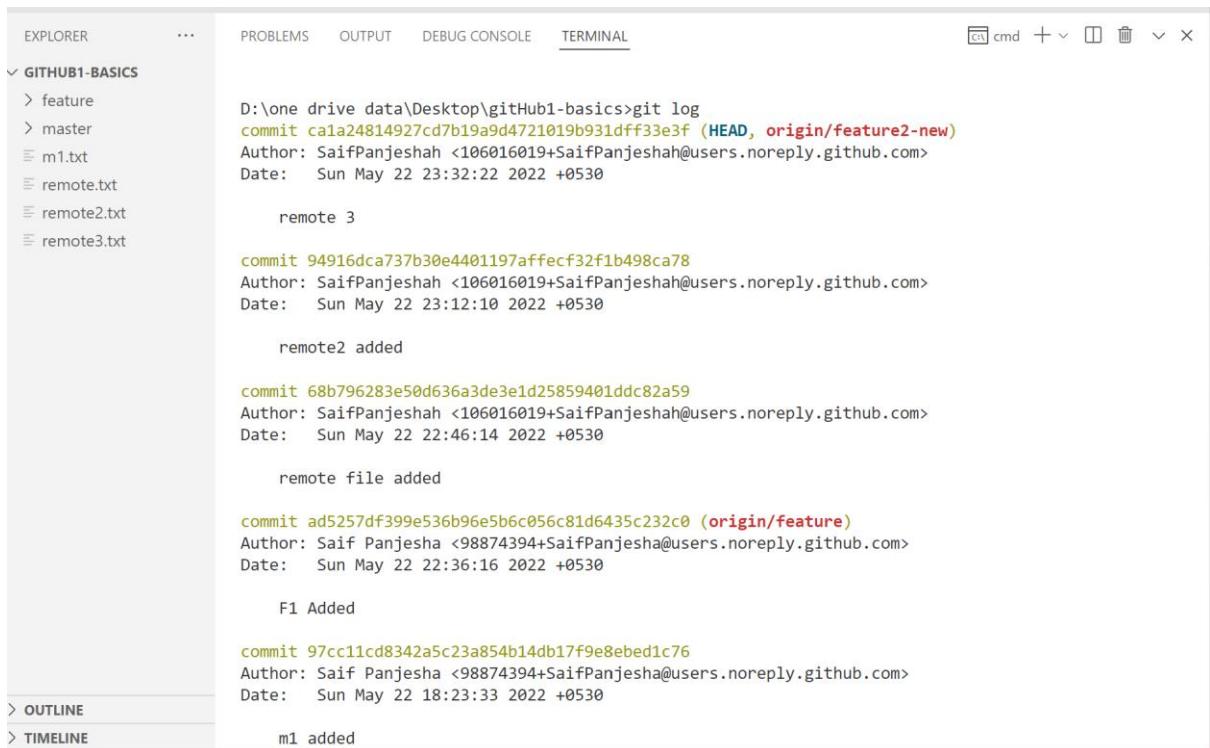
The screenshot shows the VS Code interface with the Terminal tab selected. The terminal window displays the following command and its execution:

```
D:\one drive data\Desktop\gitHub1-basics>git branch
  master
* 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 full commit history for the 'feature2-new' branch:

```
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 97cc11cd8342a5c23a854b14db17f9e8ebcd1c76
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:

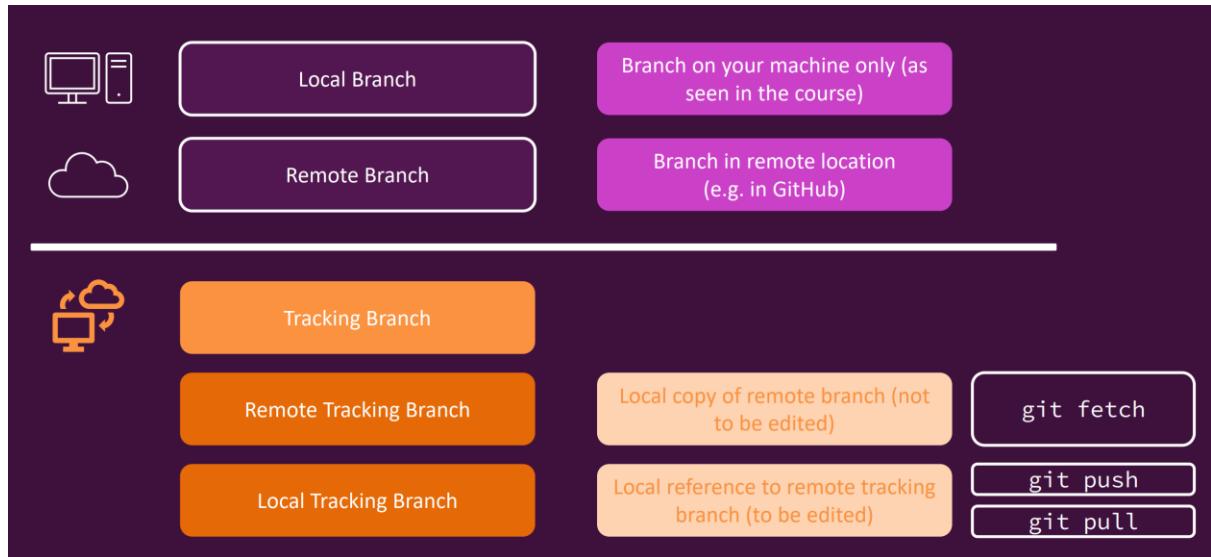


Fig. Overview of Branches

A screenshot of a terminal window showing the output of several git commands. The terminal interface includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL, with the TERMINAL tab selected. The command history shows:

```
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

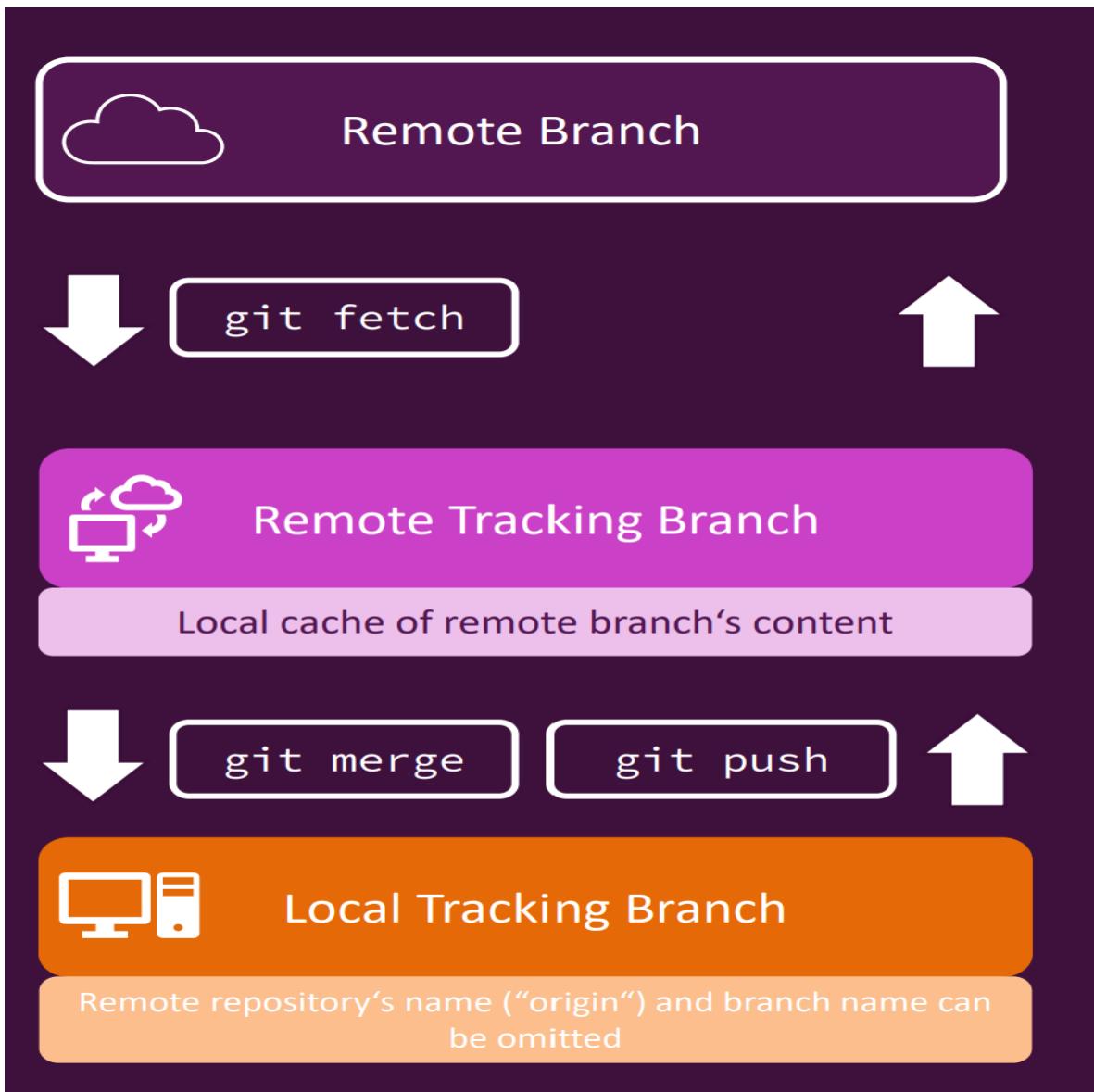


Fig. Local & remote tracking branches

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 following details:

- EXPLORER** sidebar: Shows branches feature, master, and remote tracking. Under remote tracking, files remote.txt, m1.txt, remote2.txt, and remote3.txt are listed.
- TERMINAL** tab: Displays the command-line history:
 - 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 following details:

- EXPLORER** sidebar: Shows branches feature, master, and remote tracking. Under remote tracking, files remote.txt, m1.txt, remote2.txt, and remote3.txt are listed.
- TERMINAL** tab: Displays the command-line history:
 - 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

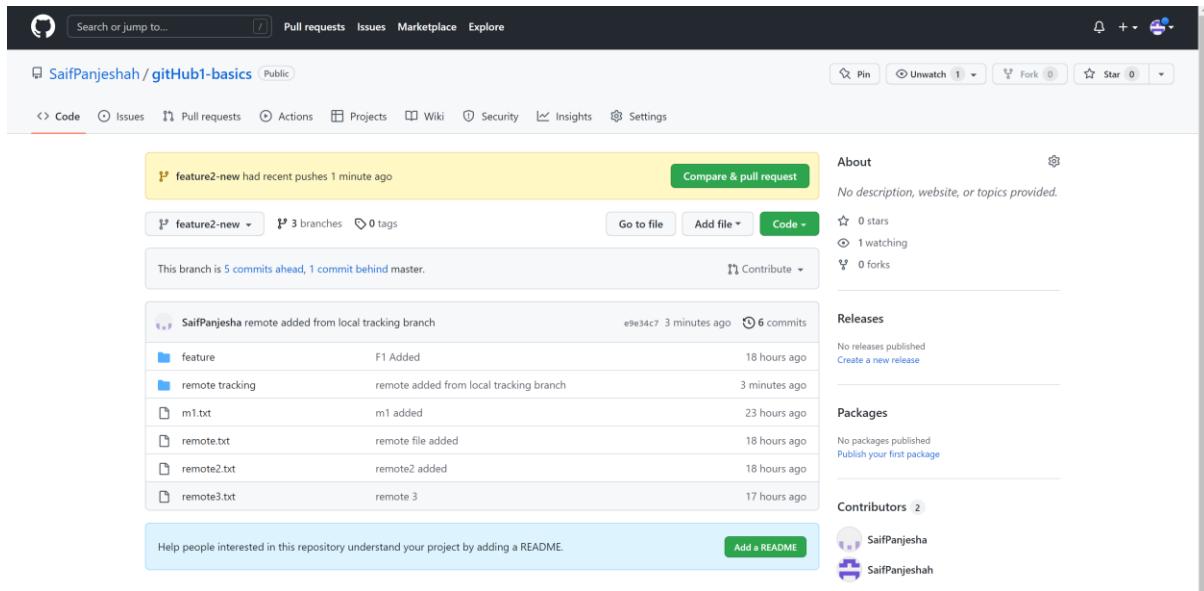


Fig. Successful on remote added from local tracking branches

Now let's try for pull from remote tracking branches to local tracking branches

Created New File :

The screenshot shows the GitHub interface for creating a new file. At the top, it says 'gitHub1-basics / remote pull branches.txt in feature2-new'. There is a 'Cancel changes' button. Below this is a text editor with a toolbar for 'Edit new file' and 'Preview'. The text area contains the number '1'. At the bottom of the editor is a 'Commit new file' dialog box. The dialog has a title 'Commit new file' and a note 'pull the branches from remote to local'. It includes a text input field for an optional extended description and two radio buttons at the bottom: one for committing directly to the 'feature2-new' branch and another for creating a new branch. There are 'Commit new file' and 'Cancel' buttons at the bottom.

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

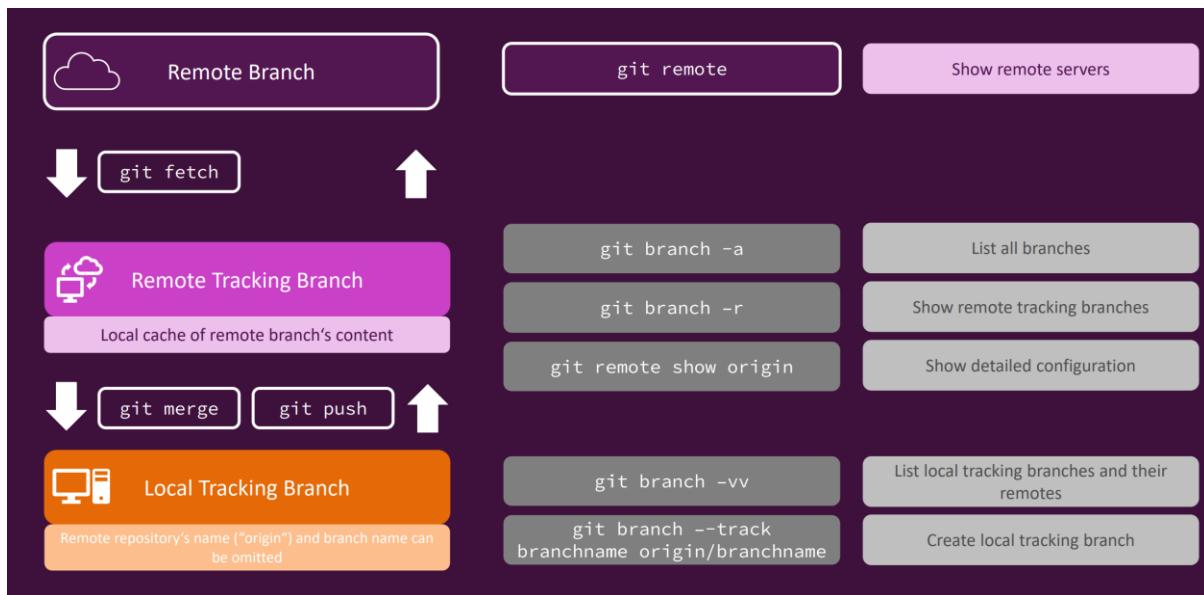


Fig. Local & Remote Tracking Branches Commands

Cloning a remote repository:

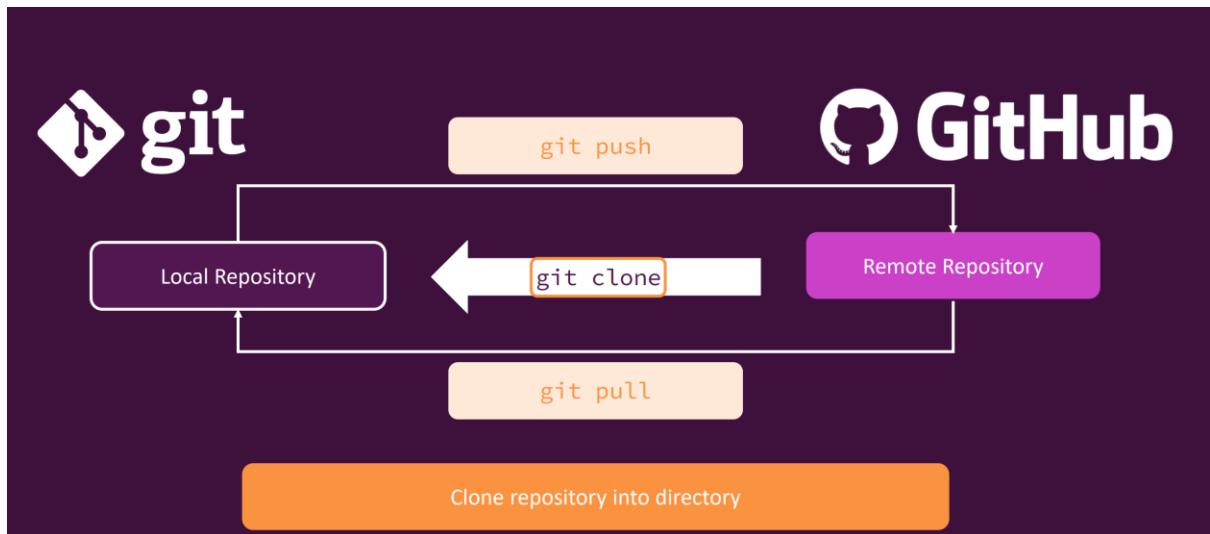


Fig. Shows How to Clone a remote repository

git clone is a git command line utility used to target an existing repository and create a clone, or copy of the target repository.

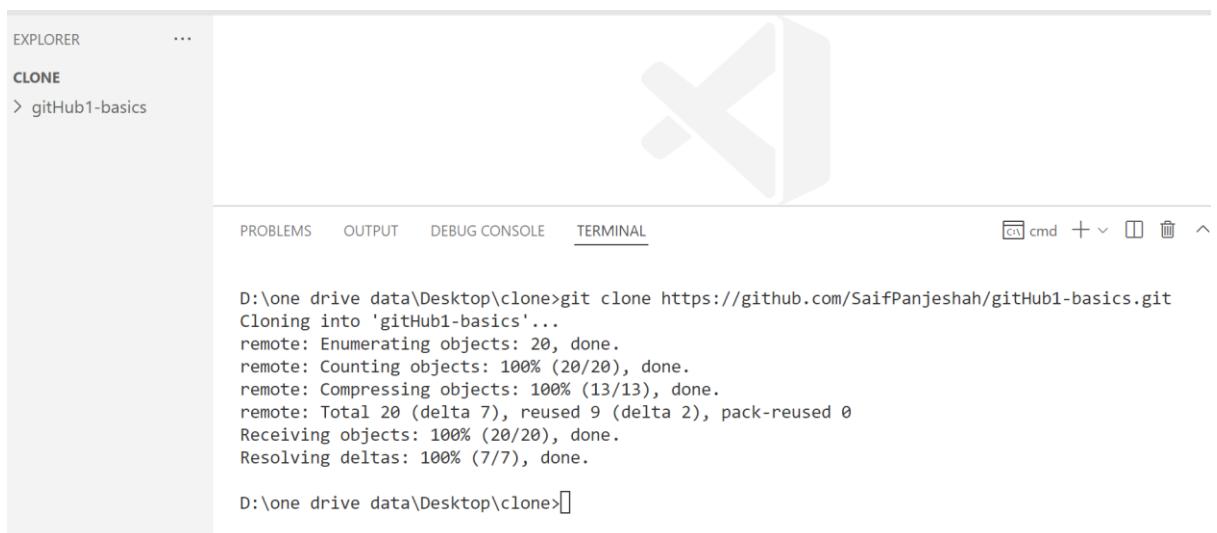
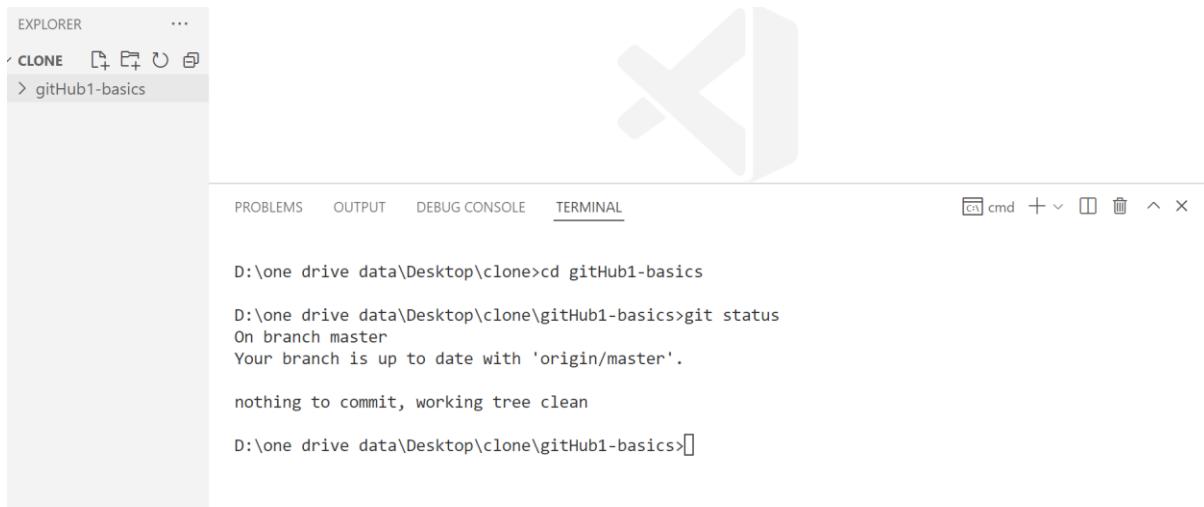


Fig. Cloning repository into folder

```
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?

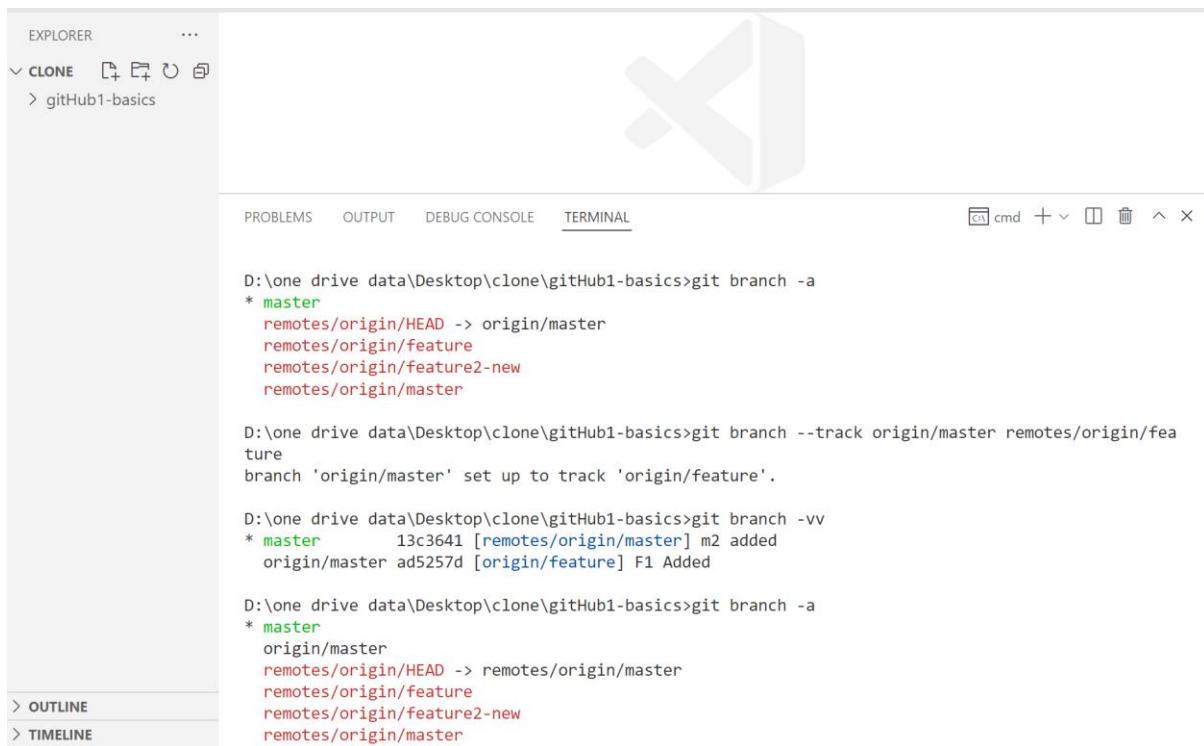


The screenshot shows the VS Code interface. In the Explorer sidebar, there is a 'CLONE' section with a single repository named 'gitHub1-basics'. The main area is a terminal window with the following text:

```
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 single repository named 'gitHub1-basics'. The main area is a terminal window with the following text:

```
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

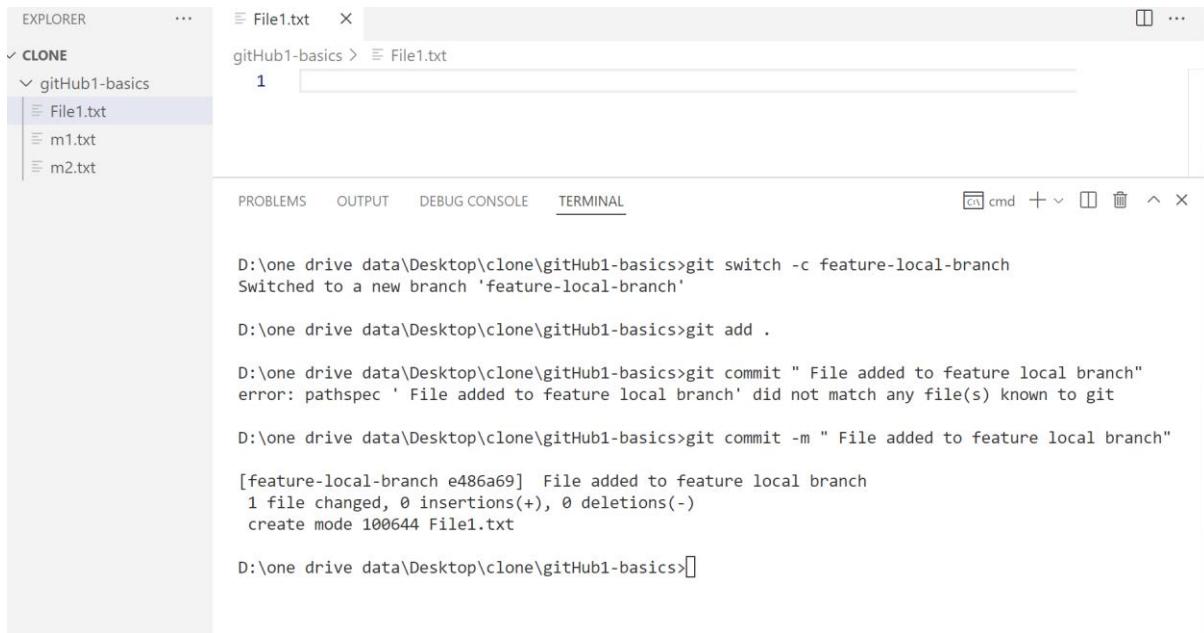
D:\one drive data\Desktop\clone\gitHub1-basics>git branch --track origin/master remotes/origin/fea
ture
branch 'origin/master' set up to track 'origin/feature'.

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 command `git switch -c feature-local-branch` is run, switching to a new local branch named 'feature-local-branch'. The terminal output also shows the commit message 'File added to feature local branch' and the file 'File1.txt' being added.

```
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 command `git push origin feature-local-branch` is run, successfully pushing the local branch to the remote repository. The terminal output shows the push process and the creation of a pull request on GitHub.

```
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

The screenshot shows a GitHub repository page for 'SaifPanjeshah / gitHub1-basics'. The repository has 4 branches and 0 tags. A recent push to the 'feature-local-branch' is highlighted. The commit details show three commits: 'File added to feature local branch' (e486a69), 'm1 added' (yesterday), and 'm2 added' (22 hours ago). The right sidebar includes sections for About, Releases, Packages, and Contributors.

Fig. Successful added on remote branch GitHub

The screenshot shows a terminal window in VS Code displaying the output of the 'git branch -vv' command. It lists several branches, including 'feature-local-branch' which is tracking 'origin/master'. Other branches shown include 'master', 'origin/master', 'remotes/origin/HEAD', 'remotes/origin/feature', 'remotes/origin/feature-local-branch', 'remotes/origin/feature2-new', and 'remotes/origin/master'.

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>
```

```
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:

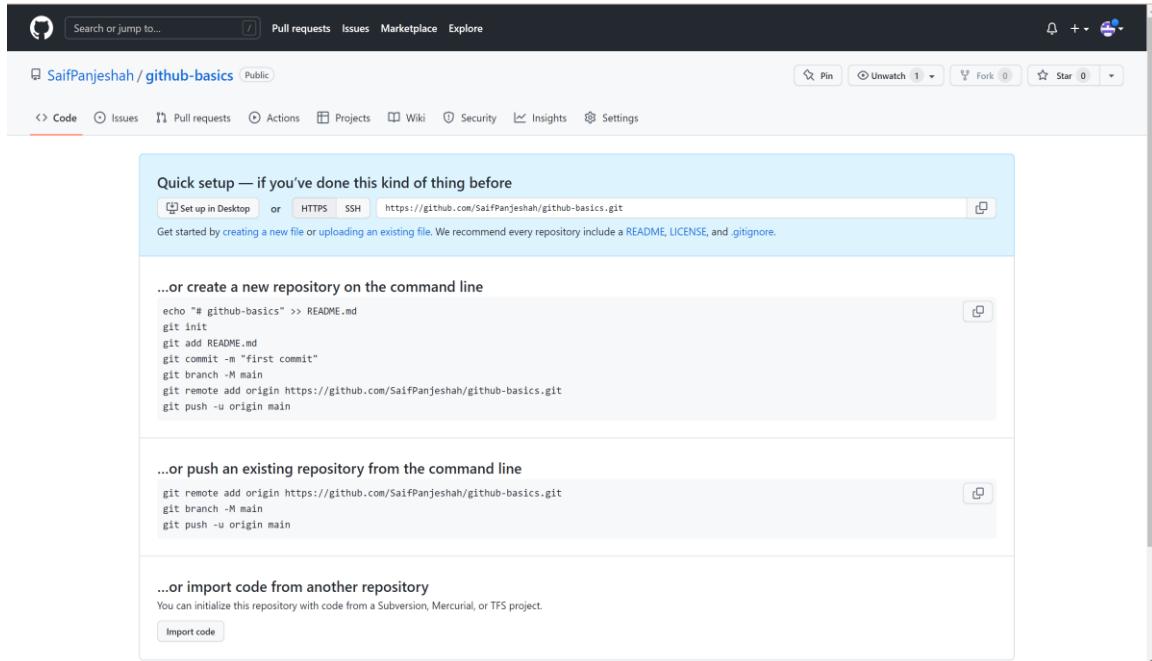


Fig. Checkout for “-u” Upstream

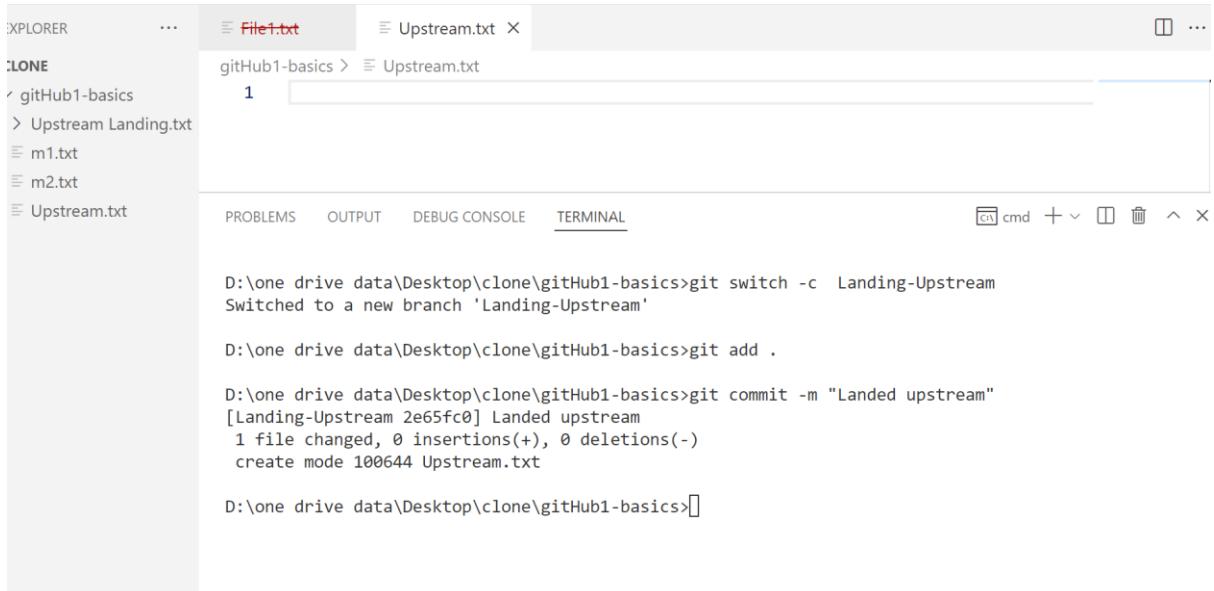


Fig. Created upstream branch

The screenshot shows the VS 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

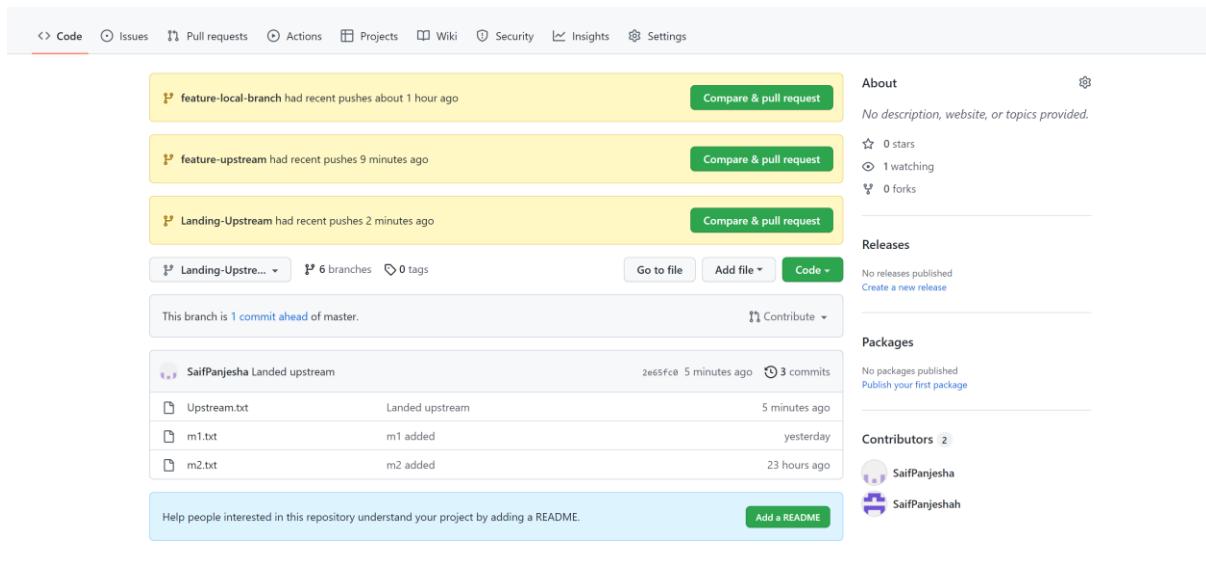
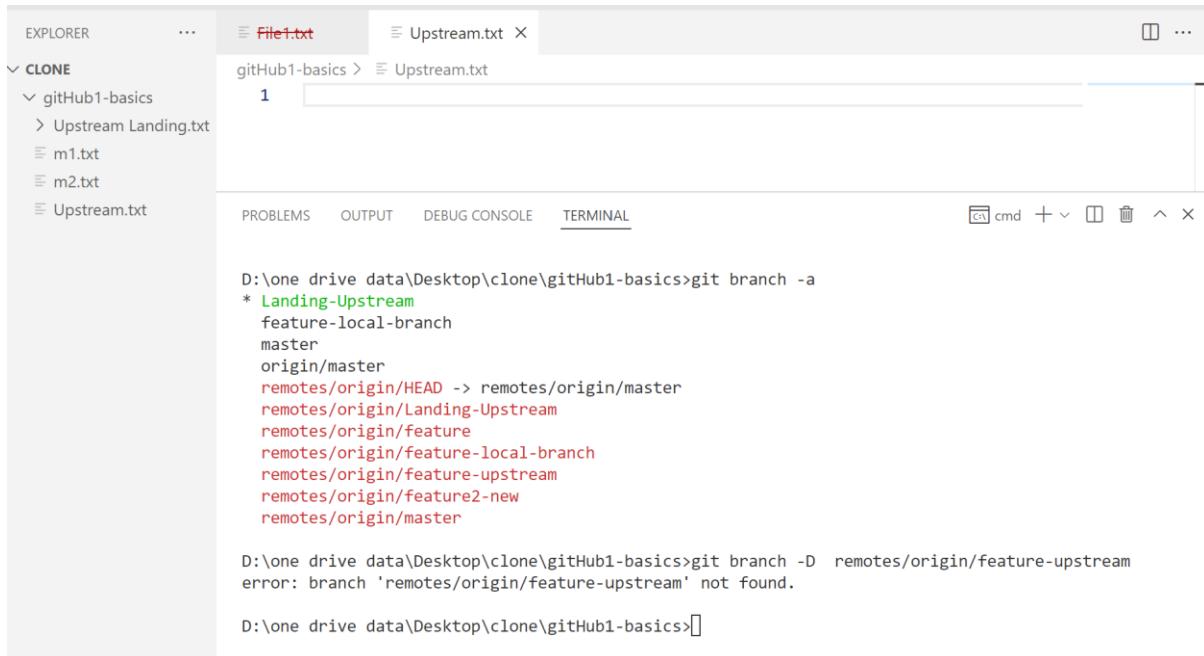


Fig. Successful Upstream.txt on GitHub

Deleting Remote Branches & Public Commits:



The screenshot shows the VS Code interface with the terminal tab active. The terminal output shows the user running the command `git branch -D remotes/origin/feature-upstream`, which results in an error message: "error: branch 'remotes/origin/feature-upstream' not found." This indicates that the branch was not successfully deleted.

```
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

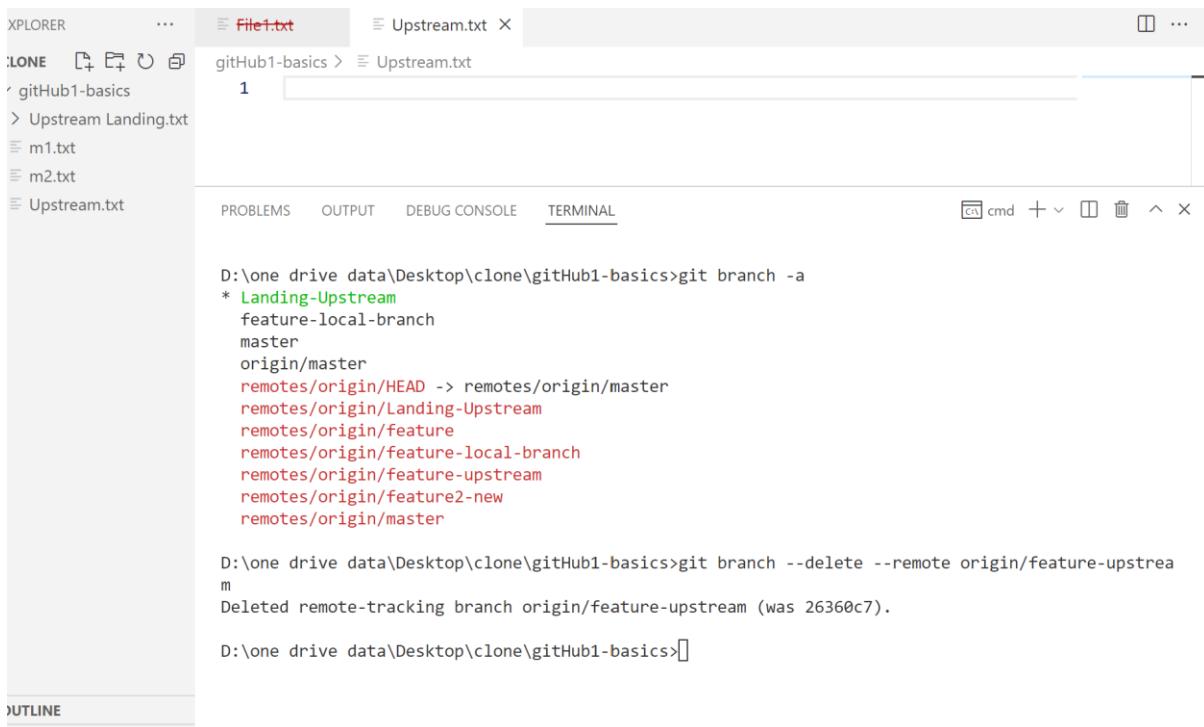
D:\one drive data\Desktop\clone\gitHub1-basics>git branch -D remotes/origin/feature-upstream
error: branch 'remotes/origin/feature-upstream' not found.

D:\one drive data\Desktop\clone\gitHub1-basics>
```

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 user running the command `git branch --delete --remote origin/feature-upstream`. The response includes the message "Deleted remote-tracking branch origin/feature-upstream (was 26360c7)." This indicates that the branch was successfully deleted.

```
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

```
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
 e486a694dbb2ac00a54696adef1a2bc3c29b722      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
 e486a694dbb2ac00a54696adef1a2bc3c29b722      refs/heads/feature-local-branch
 9e3af2165cc5f29e149eeea1207a96d2db77004b      refs/heads/feature2-new
 13c36418a467824c7c0d92cf5883dc83277fb22c      refs/heads/master
```

Fig. Successful deleted

Fig. feature-upstream successful deleted on GitHub

How to delete Public Commits:

```

XPLORER      ...      PROBLEMS      OUTPUT      DEBUG CONSOLE      TERMINAL
CLONE      ⌂ ⌂ ⌂ ⌂
gitHub1-basics
> Upstream Landing.txt
m1.txt
m2.txt

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 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>[]

```

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

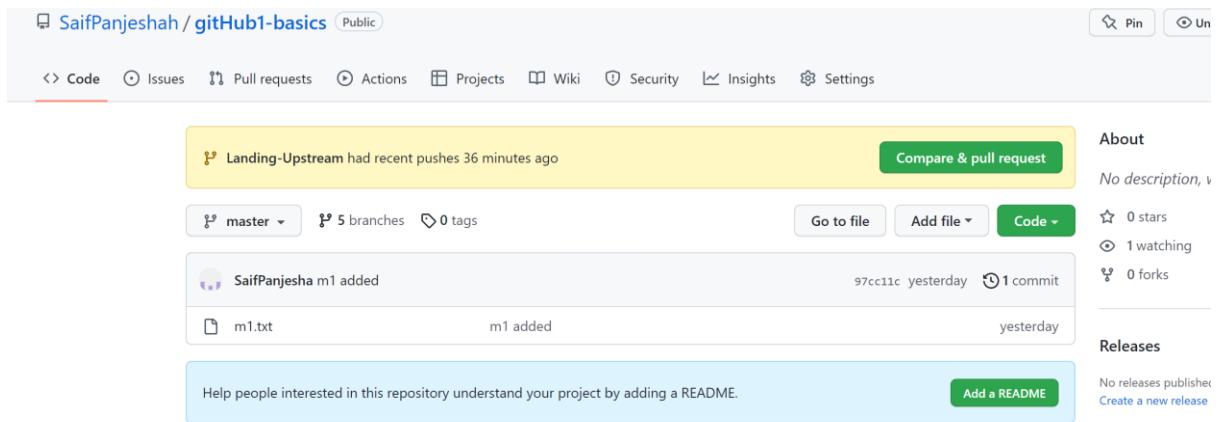


Fig. Success on Deletion GitHub

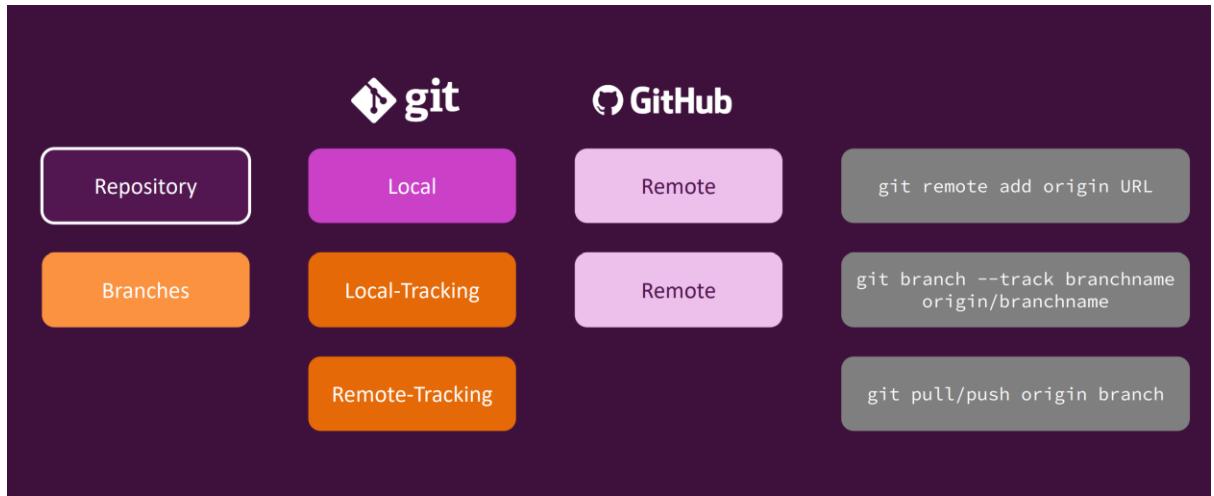


Fig. Wrap up summary

Useful Resources & Links

GitHub official website => <https://github.com/>

GitHub pricing => <https://github.com/pricing>

GitHub Deep Dive – Collaboration & Contribution

Module Introduction:

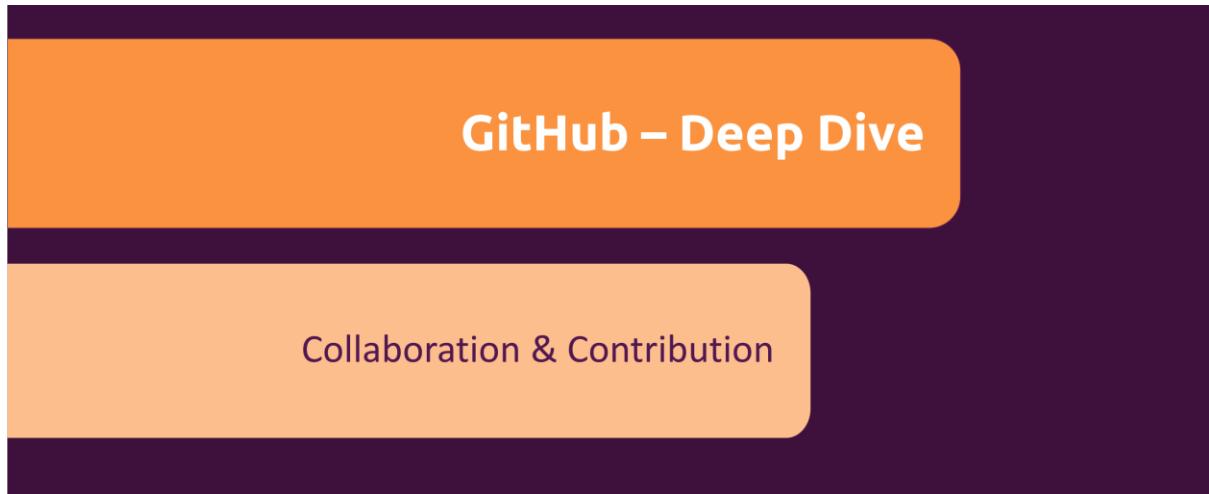


Fig. Module Introduction

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:

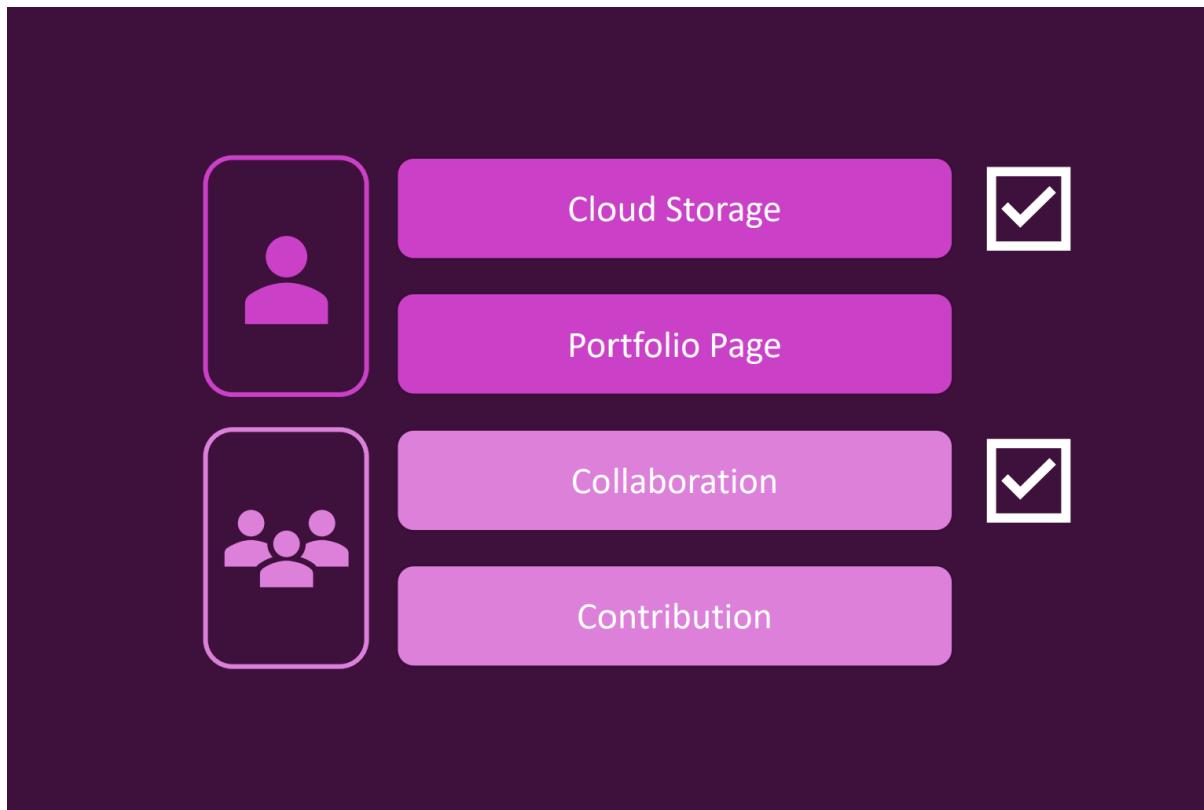


Fig. The 4 GitHub Use Cases

- **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>

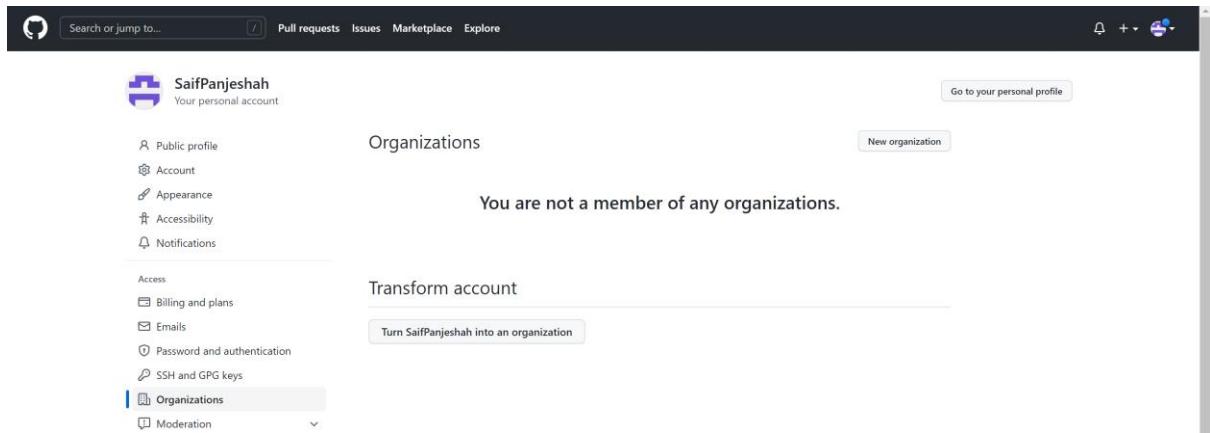


Fig. Personal GitHub Account

Changing Repository Type from Public to Private:

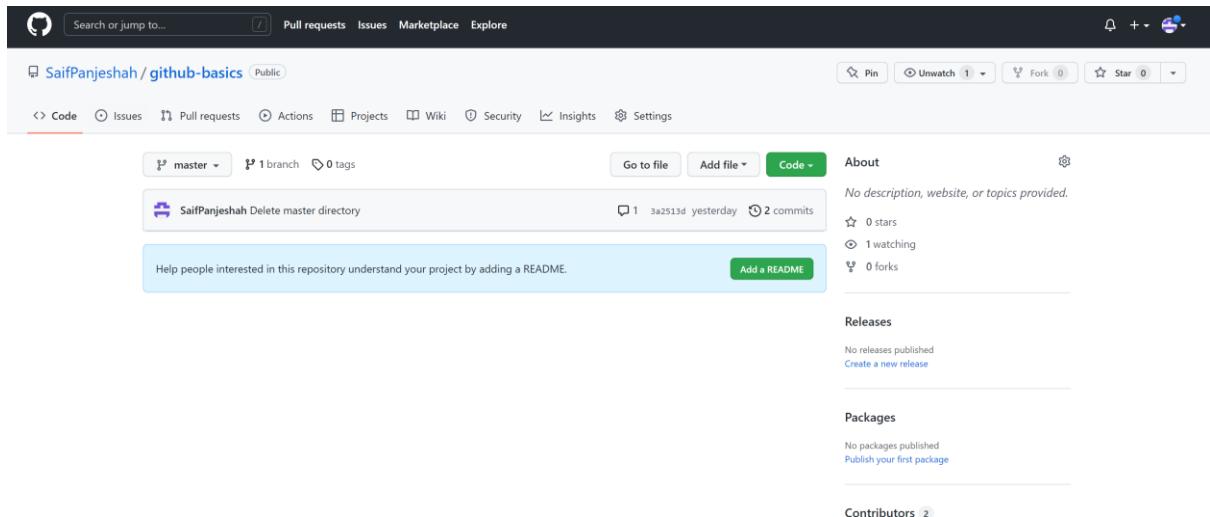


Fig. Shows Repository added in our account

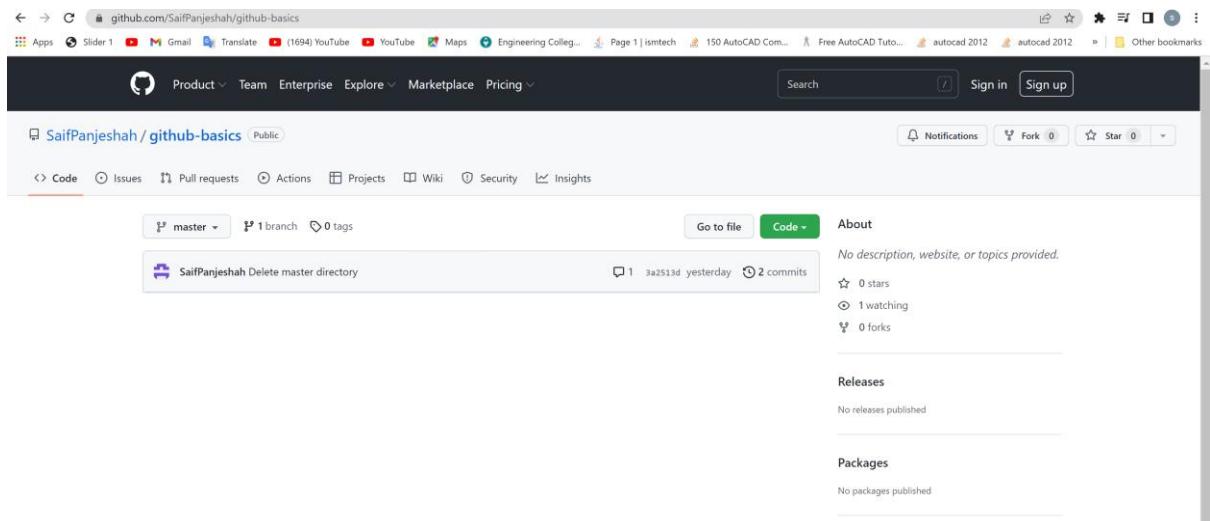


Fig. Accessing repositories from outside the browser or any user

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

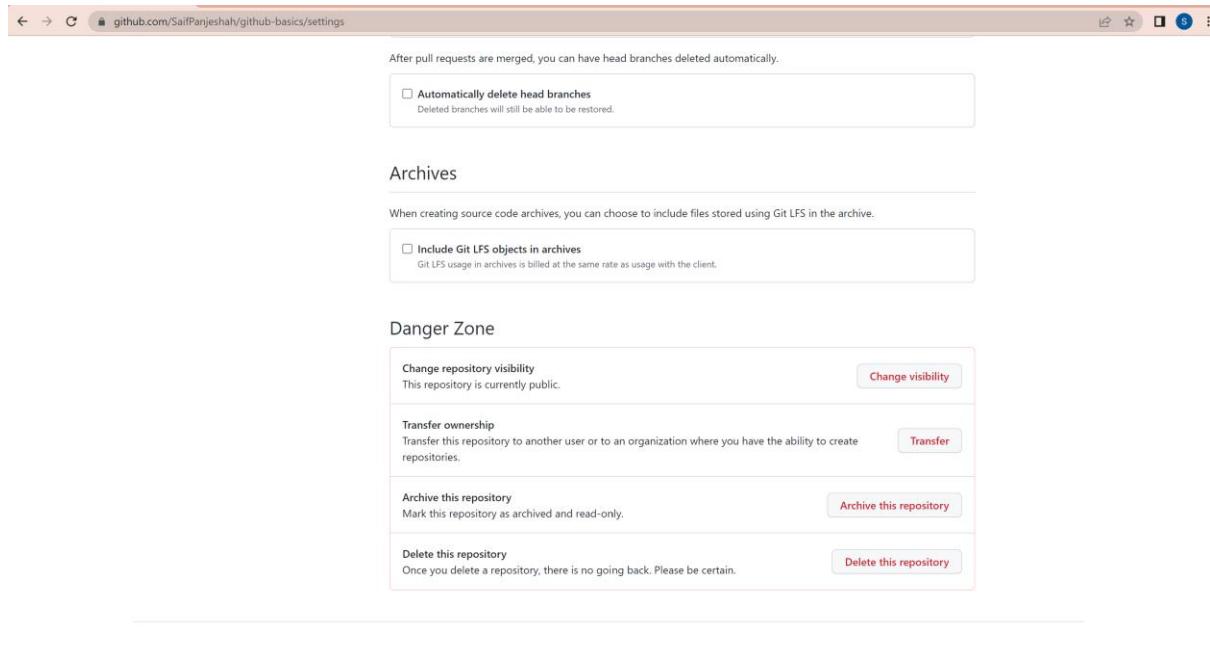


Fig. Visibility Setting of repositories

Automatically delete head branches

Delete LFS files in the archive.

Archives

When creating an archive:

Include LFS files in the archive.

Dangerous

Change repository visibility

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.

After pull requests are merged, you can have head branches deleted automatically.

Automatically delete head branches
Deleted branches will still be able to be restored.

Archives

When creating source code archives, you can choose to include files stored using Git LFS in the archive.

Include Git LFS objects in archives
Git LFS usage in archives is billed at the same rate as usage with the client.

Danger Zone

Change repository visibility
This repository is currently private. [Change visibility](#)

Transfer ownership
Transfer this repository to another user or to an organization where you have the ability to create repositories. [Transfer](#)

Archive this repository
Mark this repository as archived and read-only. [Archive this repository](#)

Delete this repository
Once you delete a repository, there is no going back. Please be certain. [Delete this repository](#)

Fig. Successful Changes from public to private repository

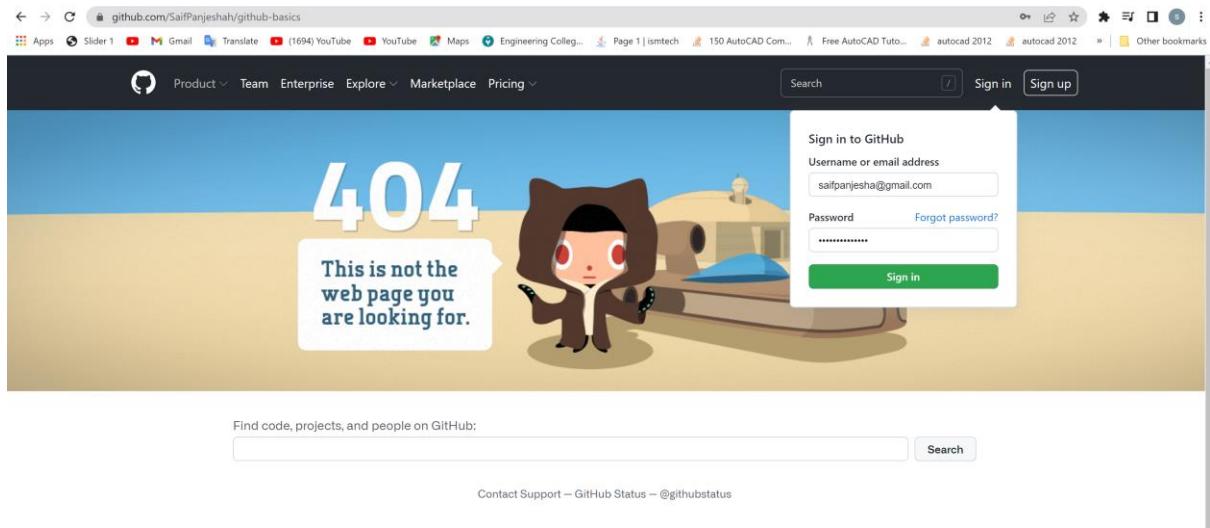


Fig. No one can access the repository

Remember: For Changing the repository public follow the Steps Vice Versa.

Pushing Commits to Public Repository

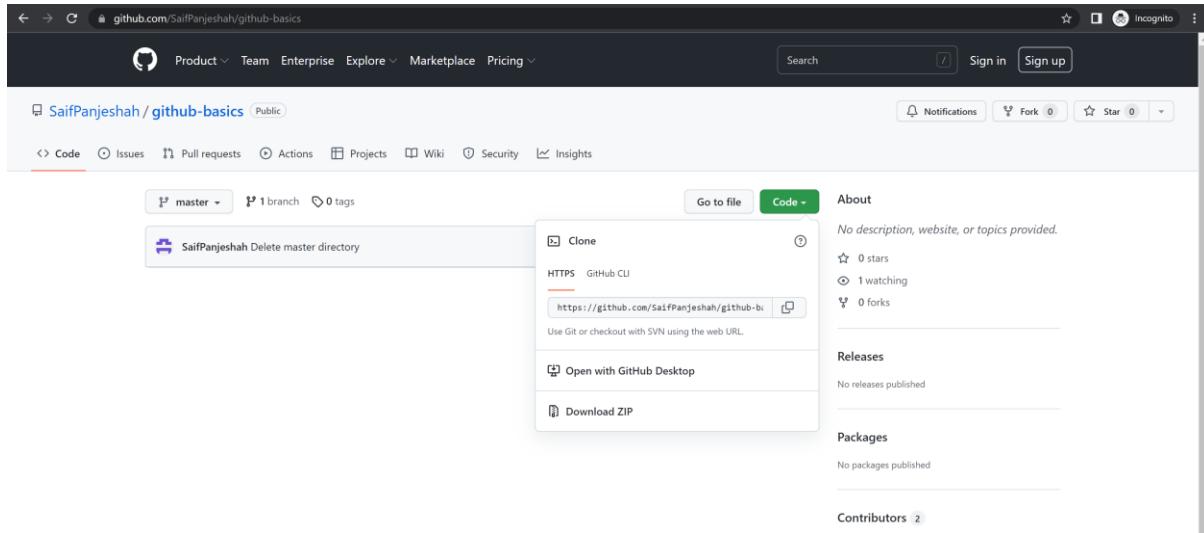
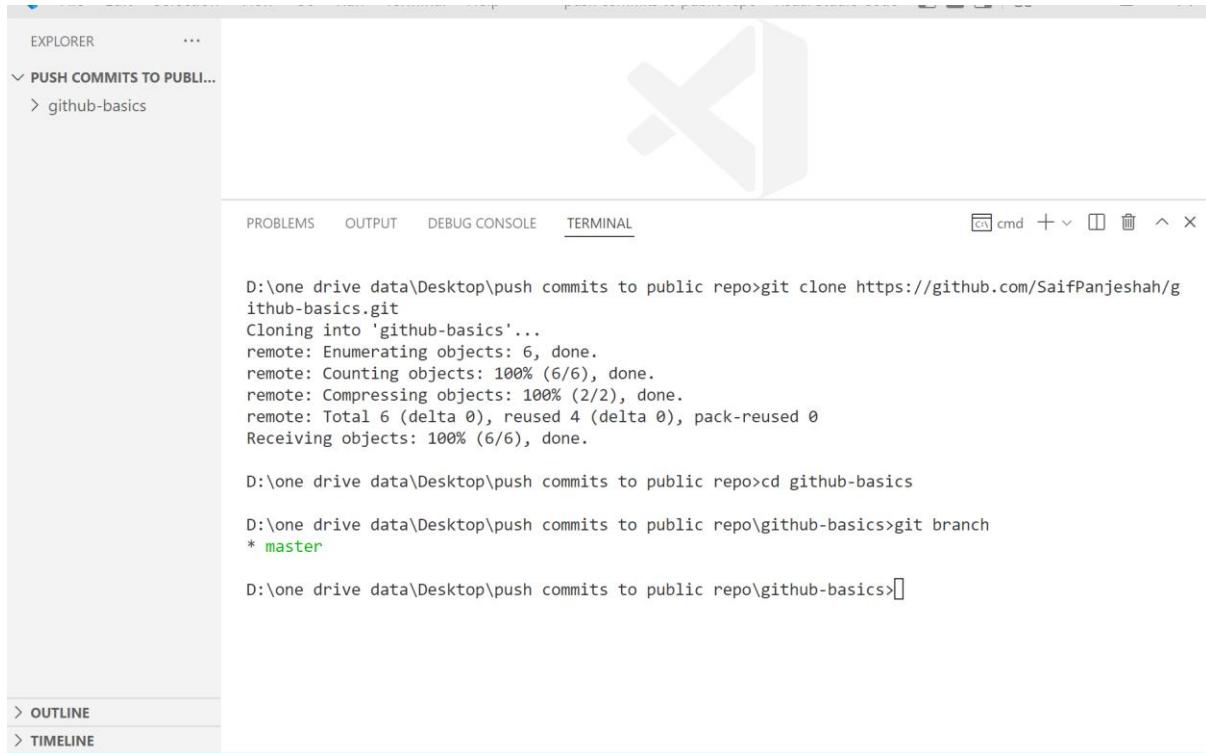


Fig. Changing the repository to public

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 main area is a terminal window 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>
```

At the bottom left of the terminal window, there are links for "OUTLINE" and "TIMELINE".

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:saiffaizalpanjesh101	Modified: Today ▾
vscodevscode.github-authentication/github.auth	Modified: 22/05/2022 ▾
MicrosoftAccount:user=saiffaizalpanjesh101@gmail.c...	Modified: Today ▾
MicrosoftAccount:user=saiffaizalpanjesh1@outlook.c...	Modified: Today ▾
OneDrive Cached Credential	Modified: Today ▾
virtualapp/didlogical	Modified: 30/04/2022 ▾

Fig. deleting all GitHub credentials

The screenshot shows the GitHub Developer settings page. The 'Personal access tokens' tab is selected. It displays a list of tokens with columns for 'Token' and 'Last used'. There are buttons for 'Generate new token' and 'Revoke all'. A note at the bottom explains that personal access tokens function like OAuth access tokens.

Fig. deleted personal access tokens

The screenshot shows the VS Code interface with a terminal window open. The terminal output shows the following command sequence:

```
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:

A screenshot of a terminal window in a code editor interface (VS Code). The terminal shows the following command history and output:

```
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 'General', 'Access' (which is selected and shows 'Collaborators'), 'Code and automation', 'Security', and 'Integrations'. The main area is titled 'Who has access' and shows two sections: 'PUBLIC REPOSITORY' (with the note 'This repository is public and visible to anyone.') and 'DIRECT ACCESS' (with the note '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 green 'Add people' button.

Fig. Adding Collaborator to a Private User Accounts

This screenshot shows the 'Who has access' section of a GitHub repository settings page. It displays two sections: 'PUBLIC REPOSITORY' (not applicable for this user) and 'DIRECT ACCESS' (showing '1 has access to this repository. 0 collaborators. 1 invitation.'). Below this is a 'Manage access' section. A pending invitation for 'SaifPanjesha' is listed, showing a checkbox, a 'Pending Invite' button, and a 'Remove' button. There are also 'Select all' and 'Type' dropdowns, and a search bar.

Fig. Pending User Invitation

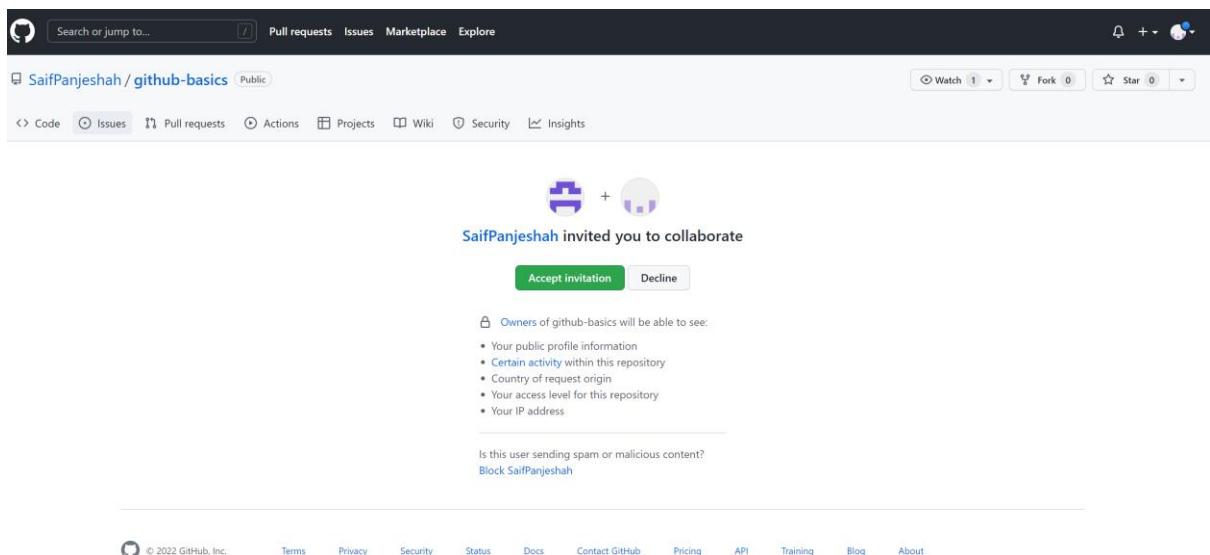


Fig. Accepting User request from another user account

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'.
Under 'PUBLIC REPOSITORY':
- A note: 'This repository is public and visible to anyone.'
- A 'Manage' button.
Under 'DIRECT ACCESS':
- A note: '1 has access to this repository. 1 collaborator.'
- A 'Type' dropdown menu.
The overall interface is light gray with blue links for 'Manage' and 'Type'.

Manage access

This screenshot shows the 'Manage access' interface. It features a search bar labeled 'Find a collaborator...' and a list of users with checkboxes:
- A checkbox next to 'Select all'.
- A search bar labeled 'Type'.
- A list item: 'SaifPanjeshah Collaborator' with a 'Remove' button to its right.
Below the list, there are navigation arrows for 'Previous' and 'Next'.

Fig. Successful added

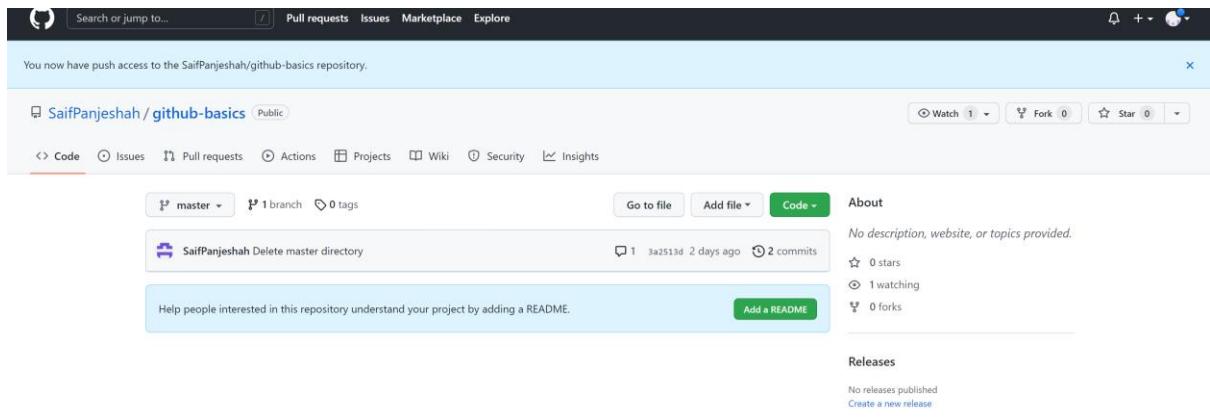


Fig. push access to another users

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)

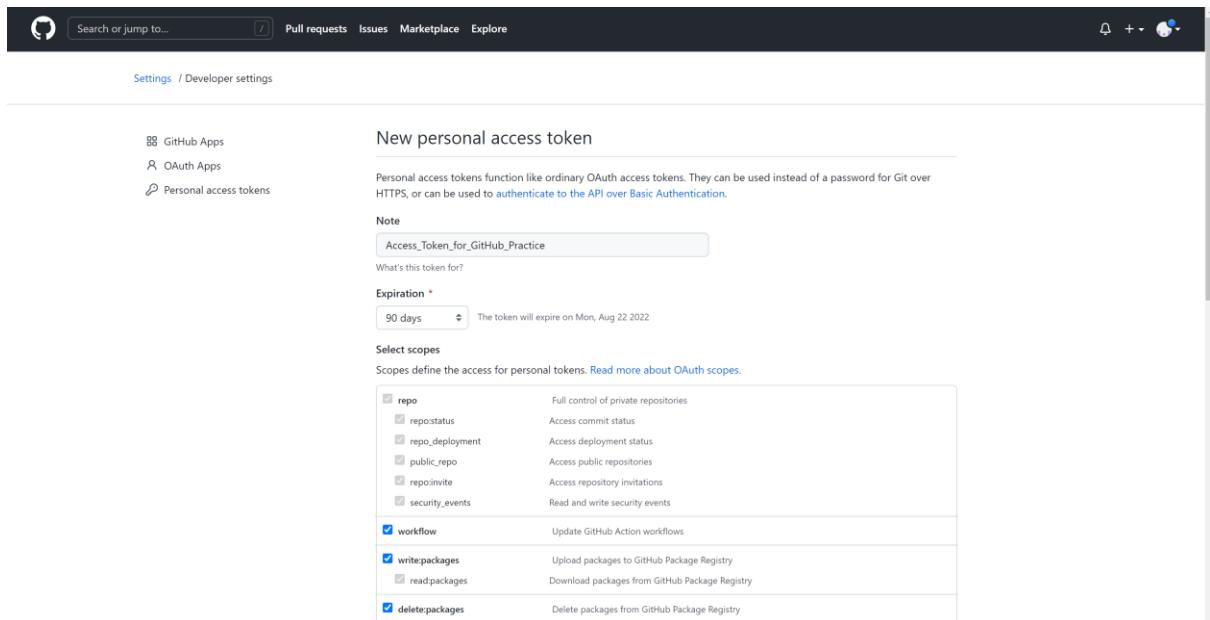


Fig. creating access Token From another user accounts

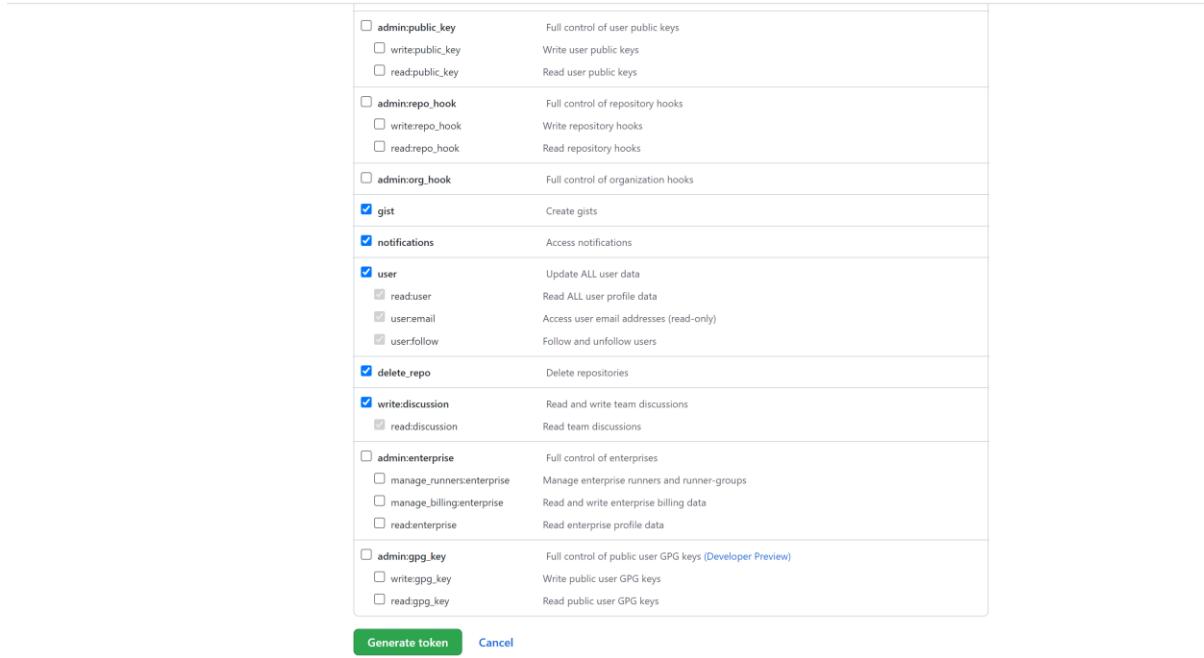


Fig. Creation Successful generate the token now without admin access

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 "Succesful push to public"
[master 670e1cb] Succesful 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

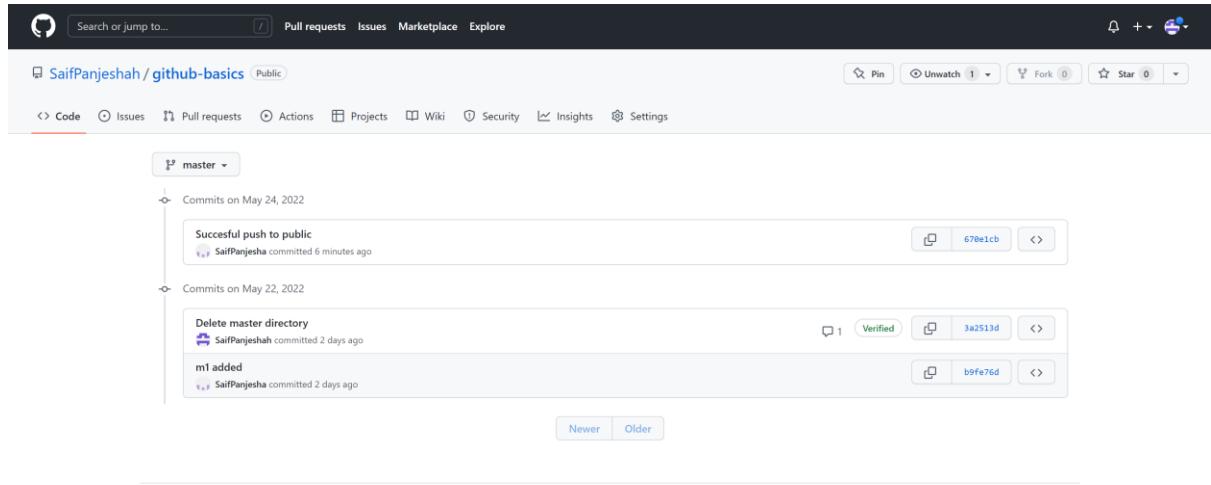


Fig. Commit Successfus added

Collaborating in Private repositories:

Danger Zone

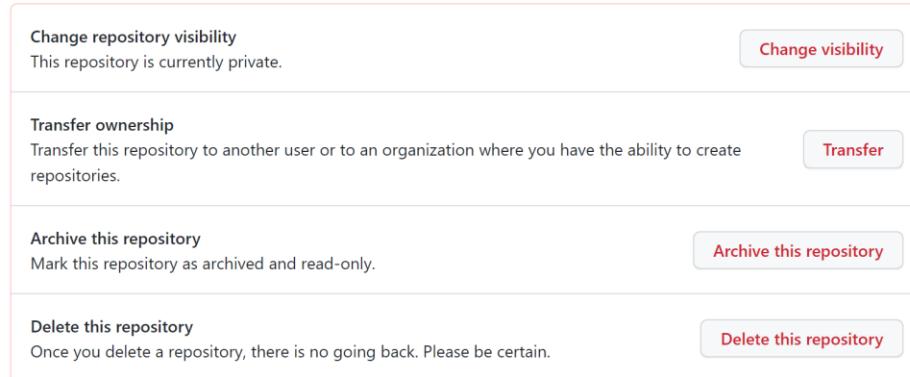


Fig. Changing repositories in private

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

- EXPLORER**: Shows a tree view with a expanded node labeled "PUSH COMMITS TO PUBLI...". Inside this node are "github-basics", "push-to-public", and "push1.txt". Below these are "file1.txt", "file2.txt", and ".gitignore".
- TERMINAL**: Displays the command-line output of a git session:


```
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 "Private push re
po!"
[master 9ee294c] Private push repo!
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 push-to-public/push1.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.
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/SaifPanjeshah/github-basics.git
    7298781..9ee294c  master -> master

D:\one drive data\Desktop\push commits to public repo\github-basics>[]
```
- OUTLINE** and **TIMELINE** buttons are visible at the bottom of the terminal area.

Fig. Successful push to private repository

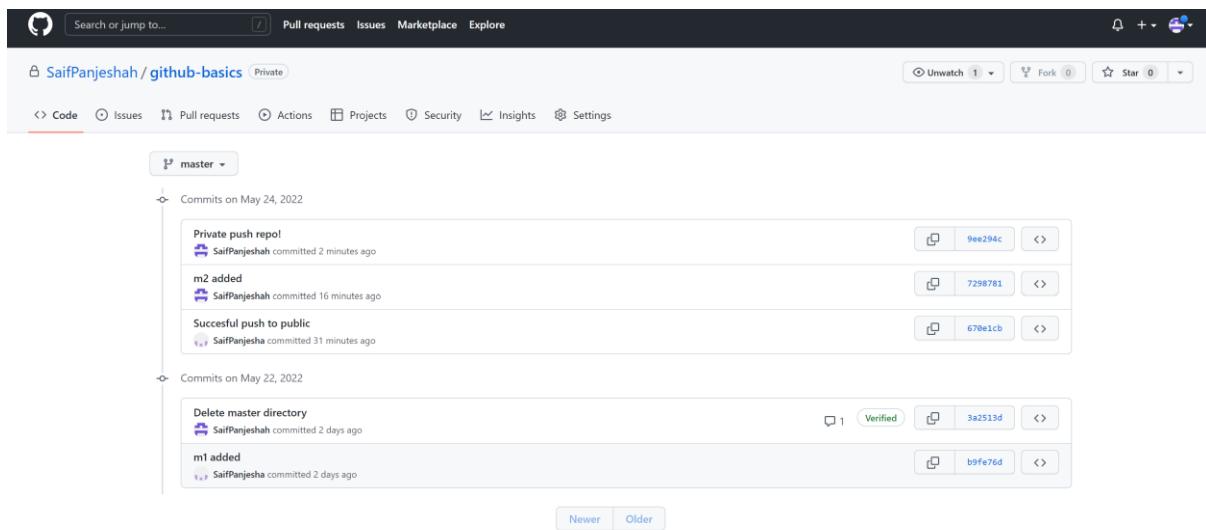


Fig. Successful push to private repository in GitHub

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 side, there's a link to 'Go to your personal profile'. The main sidebar on the left contains several sections: 'Public profile', 'Account', 'Appearance', 'Accessibility', 'Notifications', 'Access', 'Billing and plans', 'Emails', 'Password and authentication', 'SSH and GPG keys', 'Organizations', 'Moderation' (which is currently selected), 'Blocked users' (highlighted with a blue border), 'Interaction limits', and 'Code review limits'. The 'Moderation' section has a sub-section for 'Blocked users'. A message box in this section states 'You have not blocked any users.' Below this, there's a checkbox labeled 'Warn me when a blocked user is a prior contributor to a repository' with a small explanatory note: 'On repositories you haven't contributed to yet, we'll warn you when a user you've blocked has previously made contributions.' The overall interface is dark-themed.

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. It contains three sections: 'Limit to existing users', 'Limit to prior contributors', and 'Limit to repository collaborators'. Each section has checkboxes for 'New users', 'Users', 'Contributors', and 'Collaborators', with some being checked and others crossed out. There are 'Enable' dropdowns next to each section.

Fig. Limit Iteration

The screenshot shows the GitHub repository settings page for a repository named 'github-basics'. The 'General' tab is selected. The 'Repository name' is set to 'github-basics'. There is a checkbox for 'Template repository' which is unchecked. The 'Social preview' section allows uploading an image for social media preview, with a note that images should be at least 640x320px (1280x640px for best display). A 'Download template' link is also present.

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.

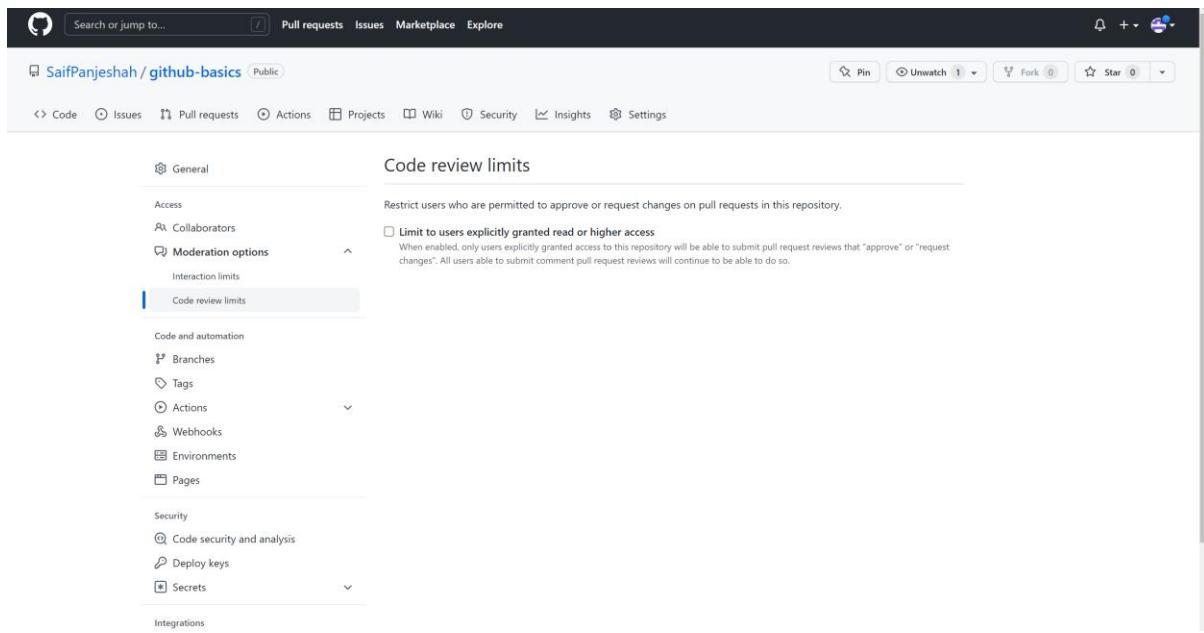


Fig. Code review limits

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

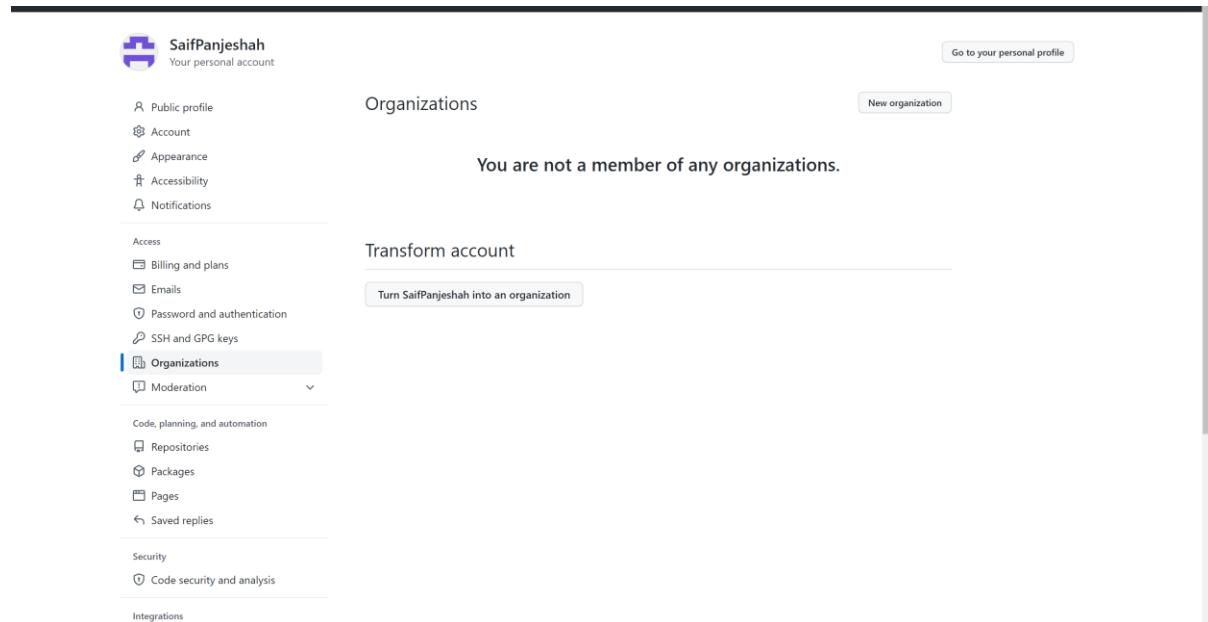


Fig. Creating organization account

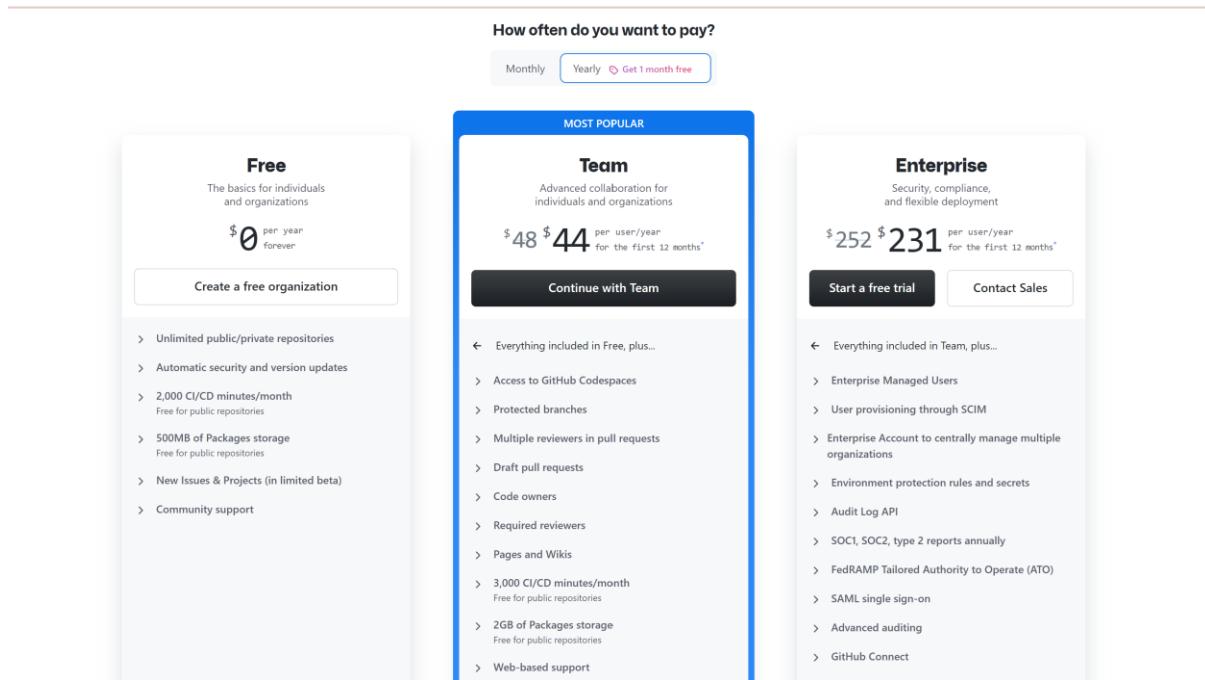


Fig. Pricing Start as Free Organization

The screenshot shows the "Create a new organization" form:

- Organization account name ***: Commitsa (highlighted with a green checkmark).
- Contact email ***: (empty input field)
- This organization belongs to: ***:
 - My personal account
Le, SaiParjeshah
 - A business or institution
For example: GitHub, Inc., Example Institute, American Red Cross
- Verify your account**: A CAPTCHA challenge asking "Please solve this puzzle so we know you are a real person".
- Acceptance checkbox**: I hereby accept the [Terms of Service](#). For more information about GitHub's privacy practices, see the [GitHub Privacy Statement](#).
- Next** button at the bottom.

Fig. Adding Organization Name Commitsa



Welcome to Commitsa

Add organization members

Organization members will be able to view repositories, organize into teams, review code, and tag other members using @mentions.

[Learn more about permissions for organizations →](#)

Search by username, full name or email address

[Complete setup](#)

[Skip this step](#)

Fig. Verified Successful

Welcome to GitHub

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.

Tell us about you

What do you spend time on most day-to-day?

Please select all that apply

- Writing code
- Managing and coordinating engineering work
- Planning projects
- Billing administration

Other

Please describe

Tell us about your team

How many people do you expect to actively work within this GitHub organization?

- 0
- 1-5
- 6-15
- 16-24
- 25+

What type of work do you plan to use this organization for?

Please select all that apply

Fig. Complete the Setup and submit

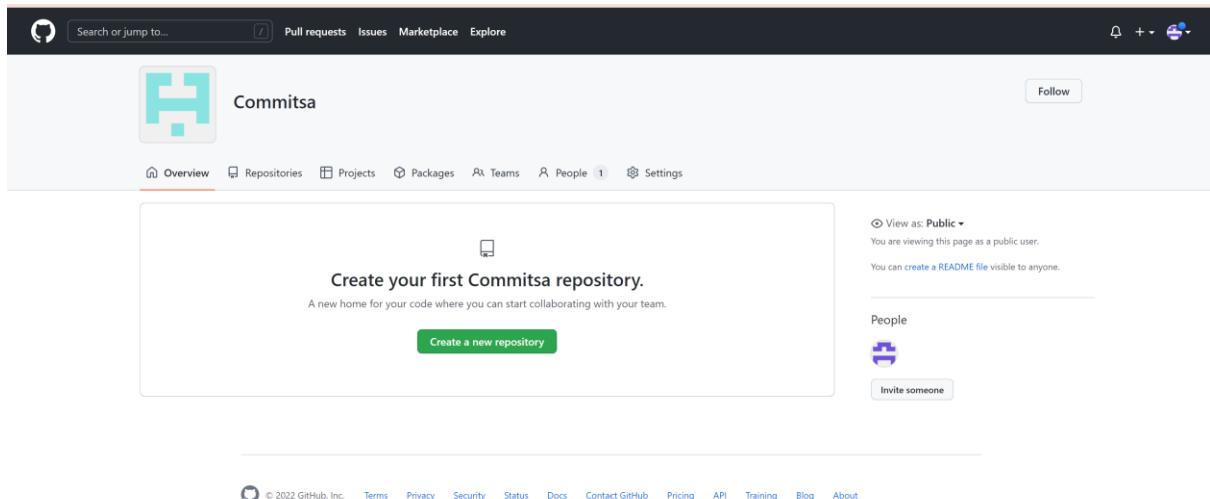


Fig. Successful Created

To View Organization from Dashboard

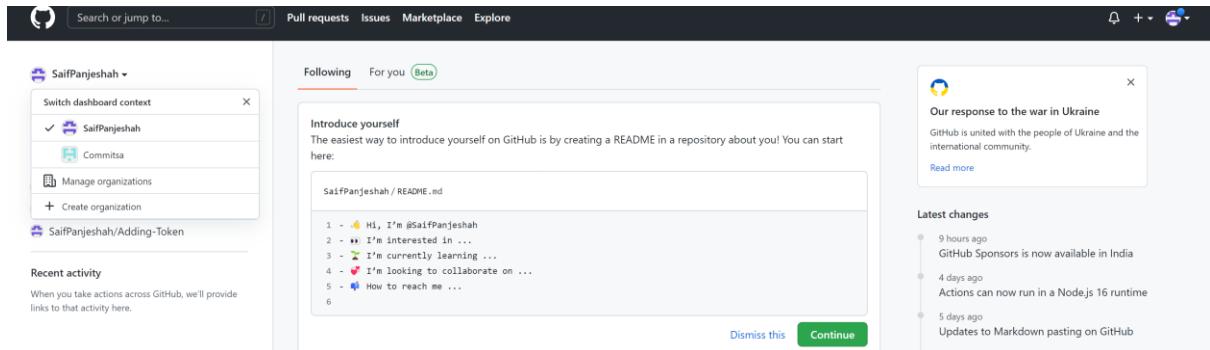


Fig. View Organization

Exploring Member Repository Permission:

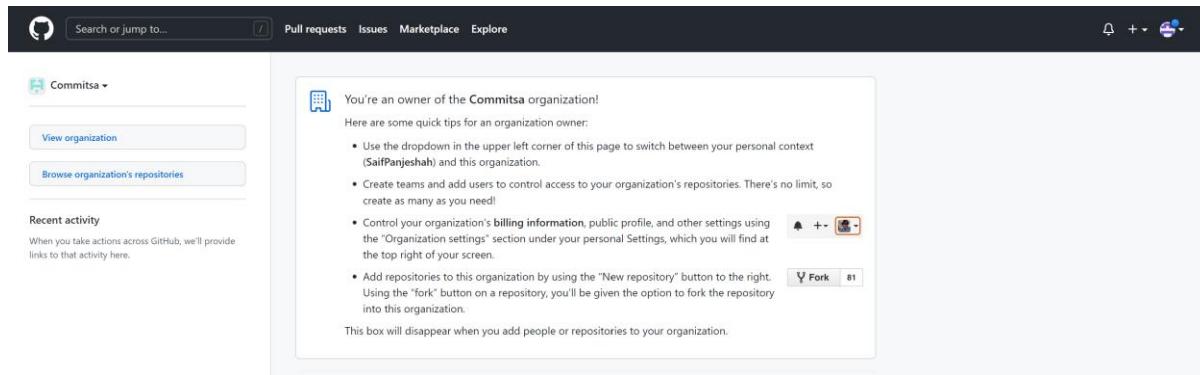


Fig. Commitsa Organization

What Missing, Here Member and repositories in Organization?

Creating New Repository:

Path: Inside Organization -> Create New Repo

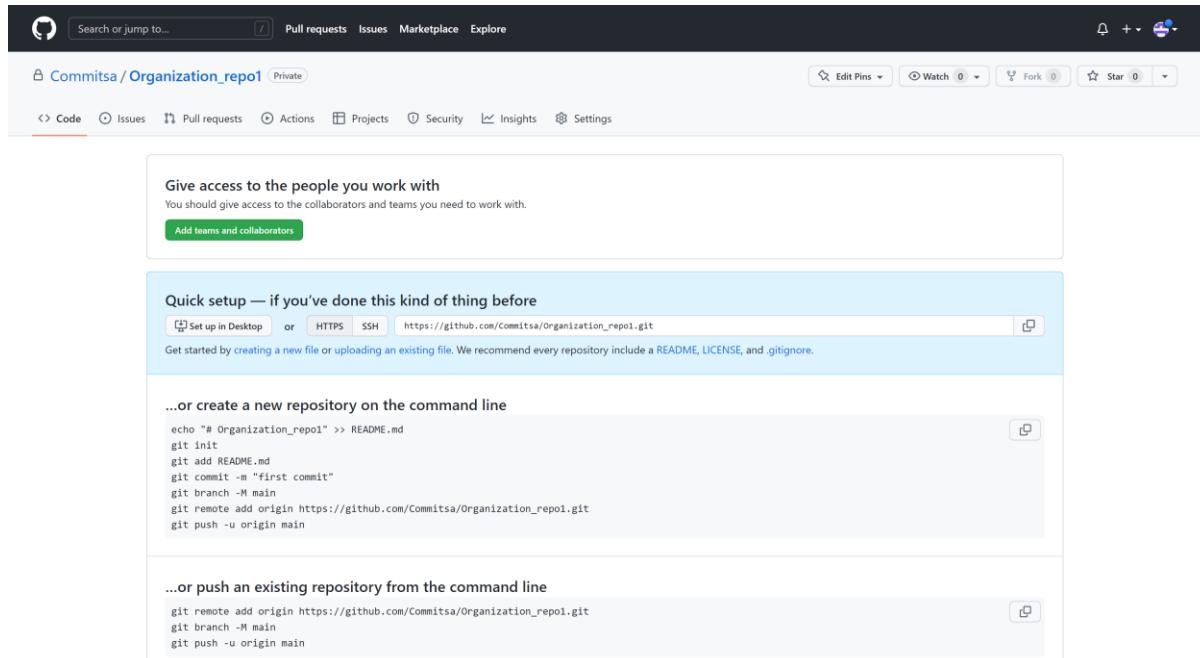


Fig. Successful Created Private Repository in Organization

After Creating repository Inviting Teams Members Same as Personal Account:

The screenshot shows the 'Settings' tab of a GitHub repository named 'Commitsa / Organization_repo1'. Under the 'Who has access' section, there are three main categories: 'PRIVATE REPOSITORY' (Only those with access to this repository can view it.), 'BASE ROLE' (All 1 members can access this repository. Read), and 'DIRECT ACCESS' (0 teams or members have access to this repository. Only Owners can contribute to this repository.). On the left sidebar, under 'Access', the 'Collaborators and teams' option is selected. In the 'Manage access' section, it says 'You haven't added any teams or people yet' and provides links to 'Add people' and 'Add teams'.

Fig. Manage access

We can privilege our base role member (account owner)

The screenshot shows the 'Member privileges' section of the GitHub organization account settings for 'Commitsa'. Under 'Base permissions', it shows that 'Public' is disabled and 'Private' is selected. Under 'Repository creation', 'Public' is disabled and 'Private' is selected. Under 'Repository forking', 'Allow forking of private repositories' is disabled. Under 'Pages creation', 'Public' is selected. A 'Save' button is visible at the bottom of each section.

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.

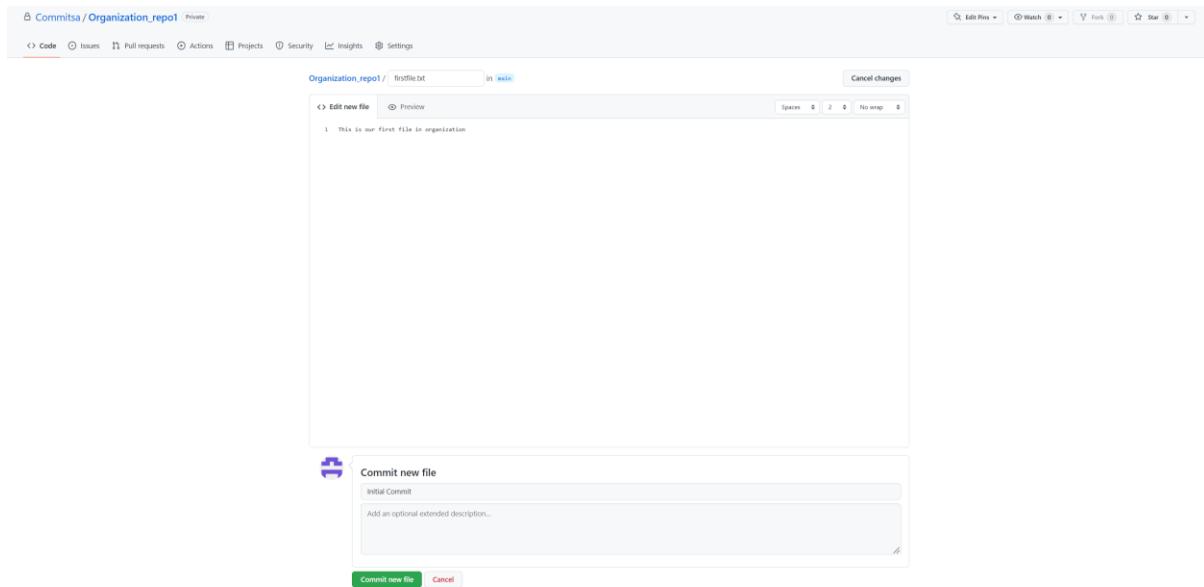


Fig. Adding First File in Organizational Account

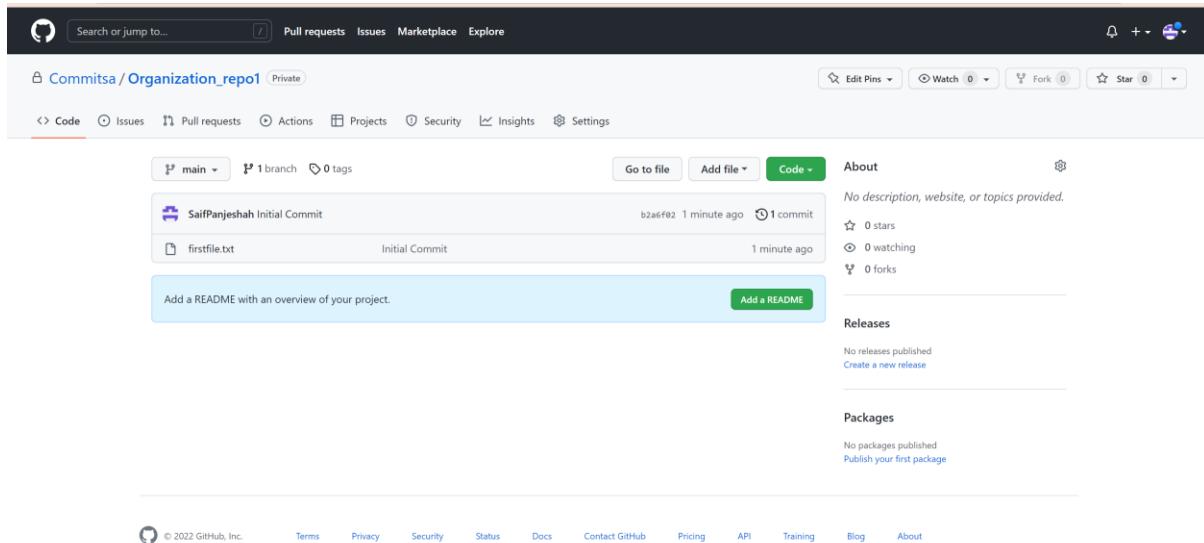


Fig. FirstFile.txt Created

The screenshot shows the GitHub repository settings page for 'Organization_repo1'. The 'Access' section is selected, specifically the 'Collaborators and teams' tab. On the left, there's a sidebar with various repository settings like 'Code and automation', 'Security', and 'Integrations'. The main area is titled 'Who has access' and shows a 'PRIVATE REPOSITORY' with a 'Read' base role assigned to 'All 1 members'. It also shows 'DIRECT ACCESS' with 0 teams or members having access. A 'Manage' button is available for both sections. Below this, a 'Manage access' section displays a message: 'You haven't added any teams or people yet' with a note about organization owners managing access. It includes 'Add people' and 'Add teams' buttons.

Fig. Collaborators and teams settings

This screenshot shows the same GitHub repository settings page as above, but with a modal window open over it. The modal is titled 'Add people to Organization_repo1' and contains a search bar with placeholder text 'Search by username, full name, or email' and a green button labeled 'Select a member above'. The background repository settings page is partially visible, showing the 'Who has access' section with the 'Collaborators and teams' tab selected. The modal obscures the 'Manage access' section below it.

Fig. Adding Outside Collaborators Organization read only permission

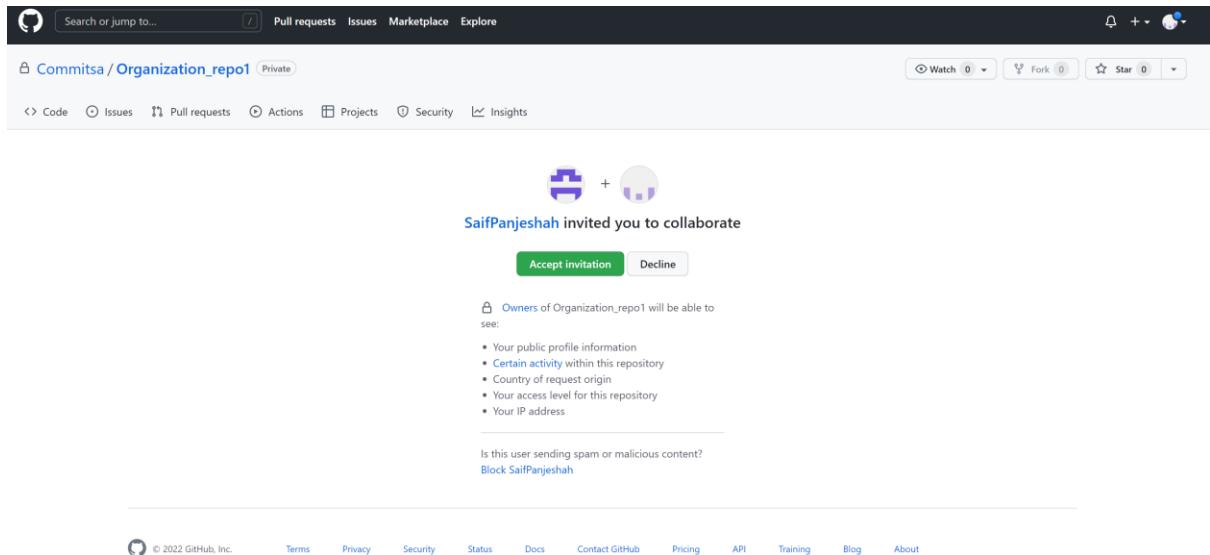


Fig. Outside Collaborators Organization Invitation

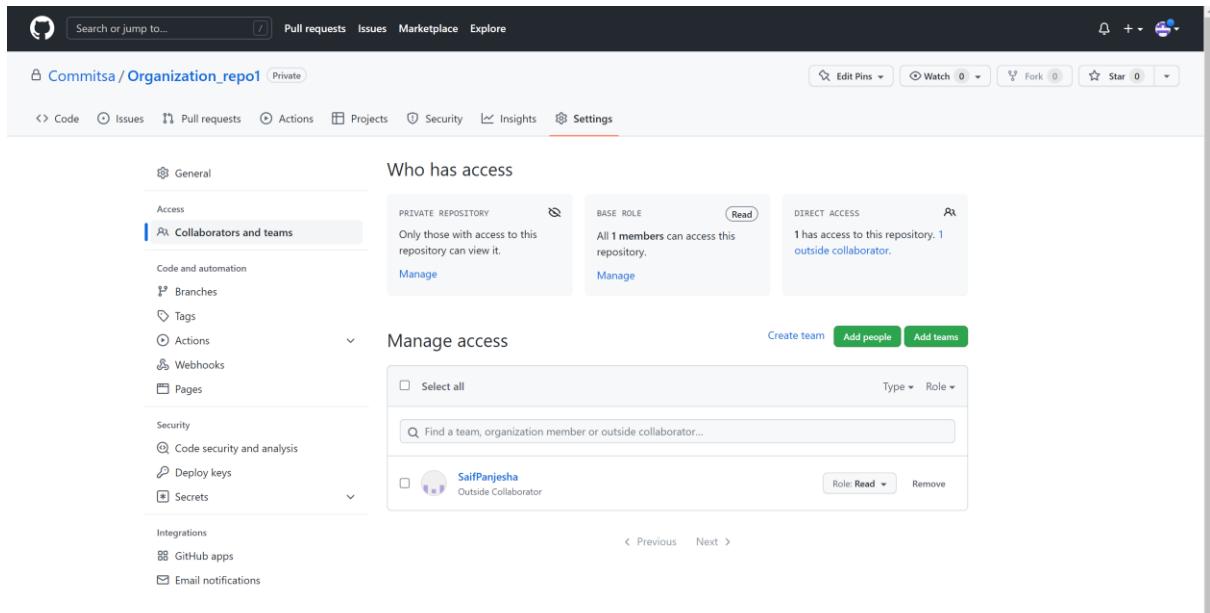


Fig. Successful added outside collaborator

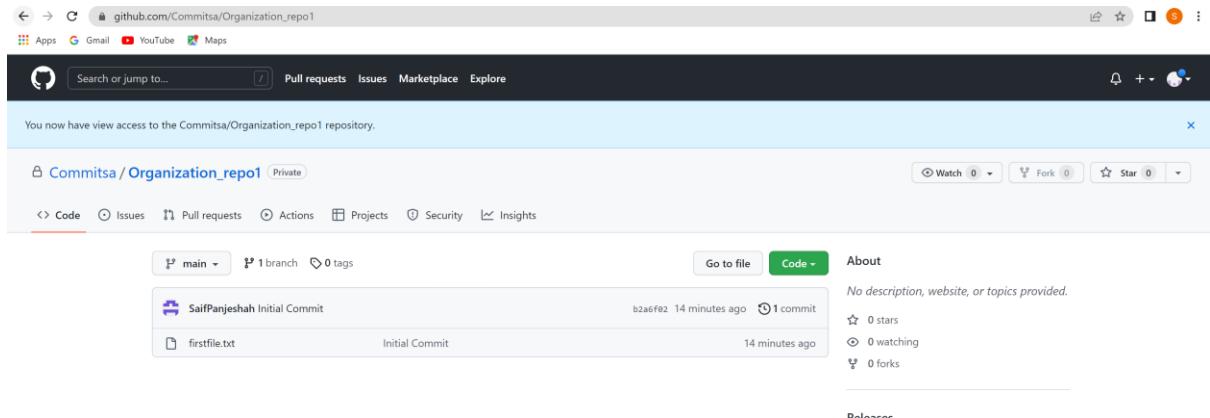


Fig. Given view access permission only

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 .

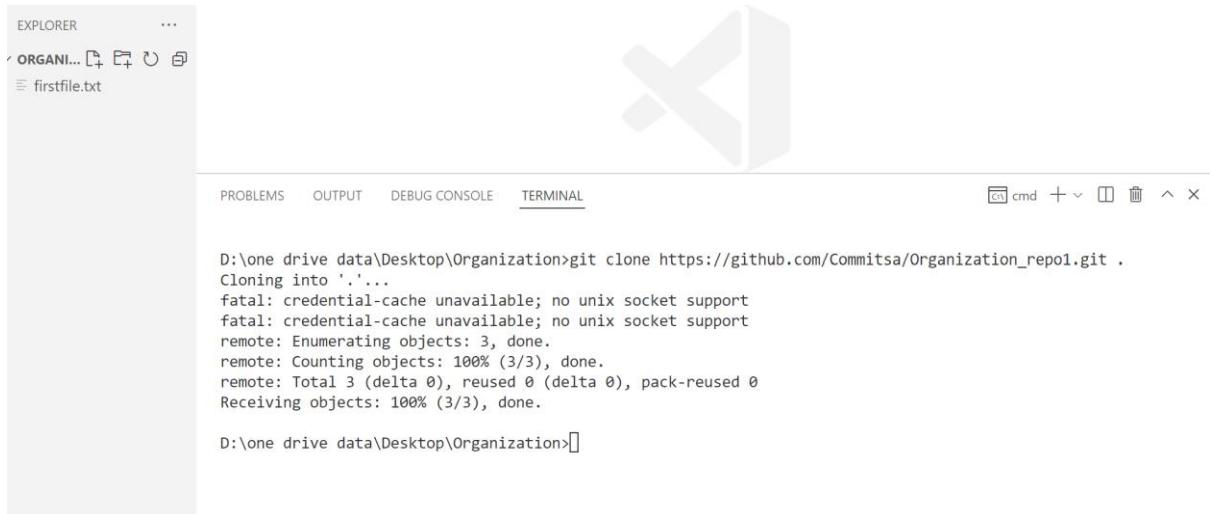


Fig. Successful created organization files

```

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

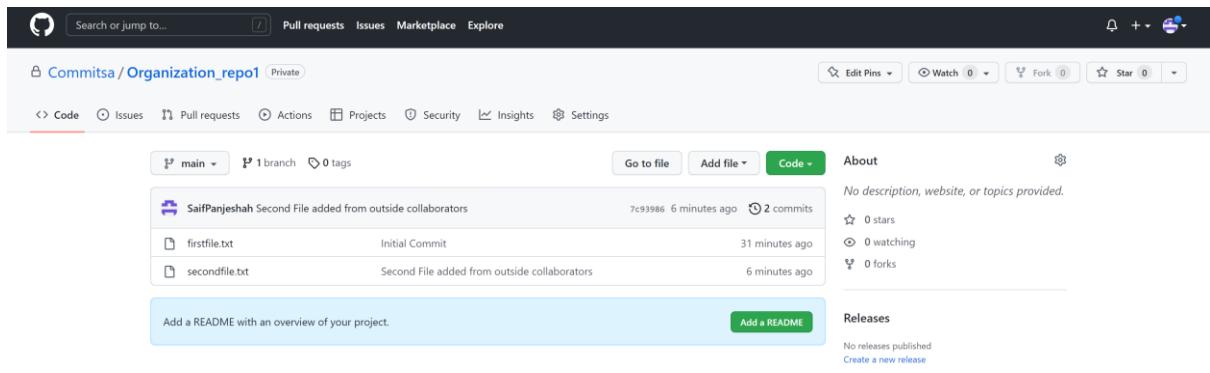


Fig. Successful added to GitHub

Adding Organization Members:

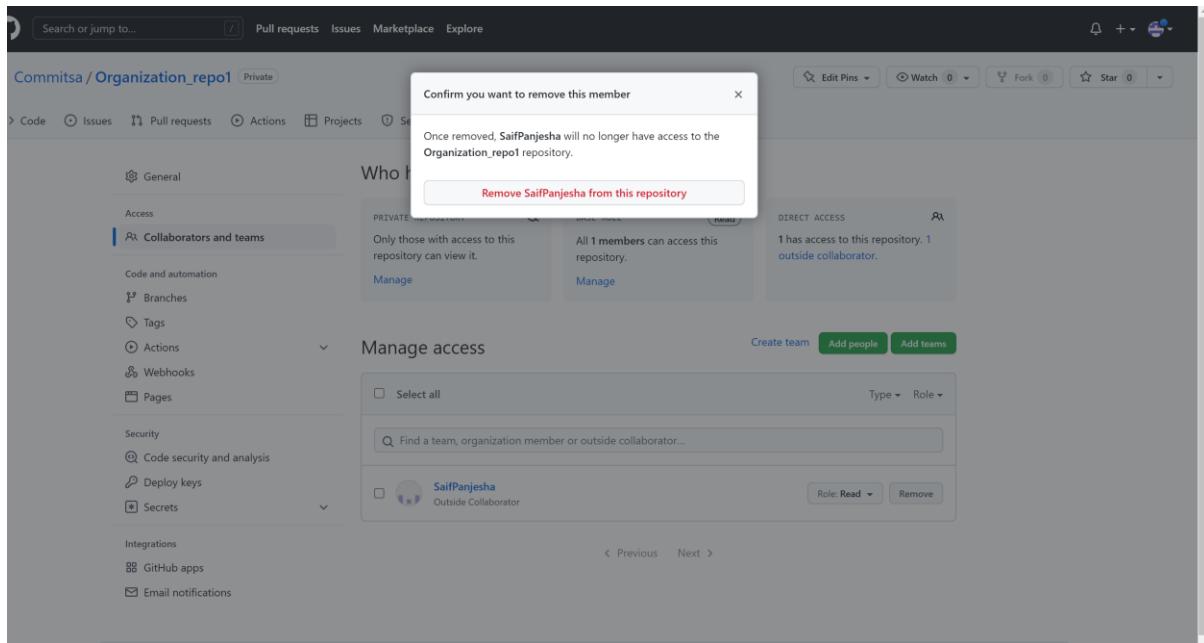


Fig. removing the outside collaborators

Let's explore people tab on Organization:

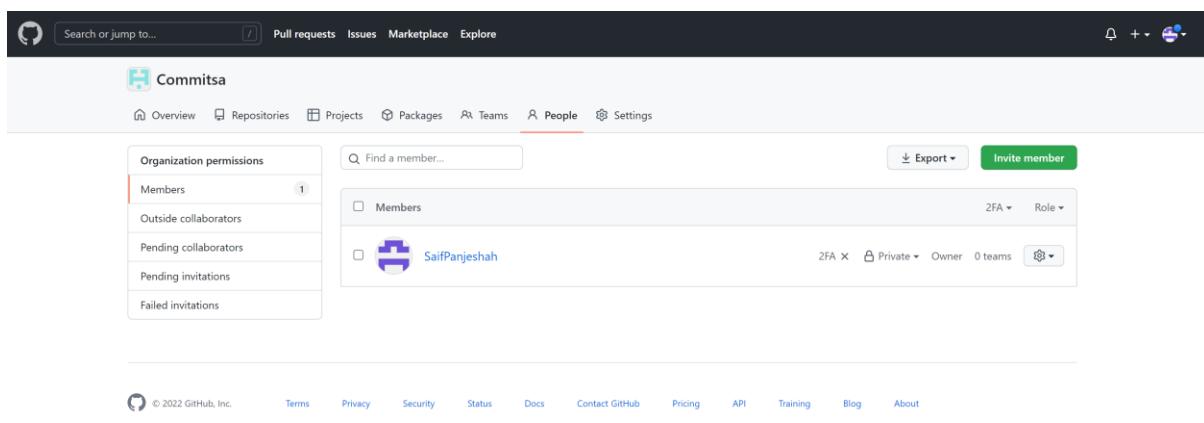


Fig. Explore people tab

Inviting Member:

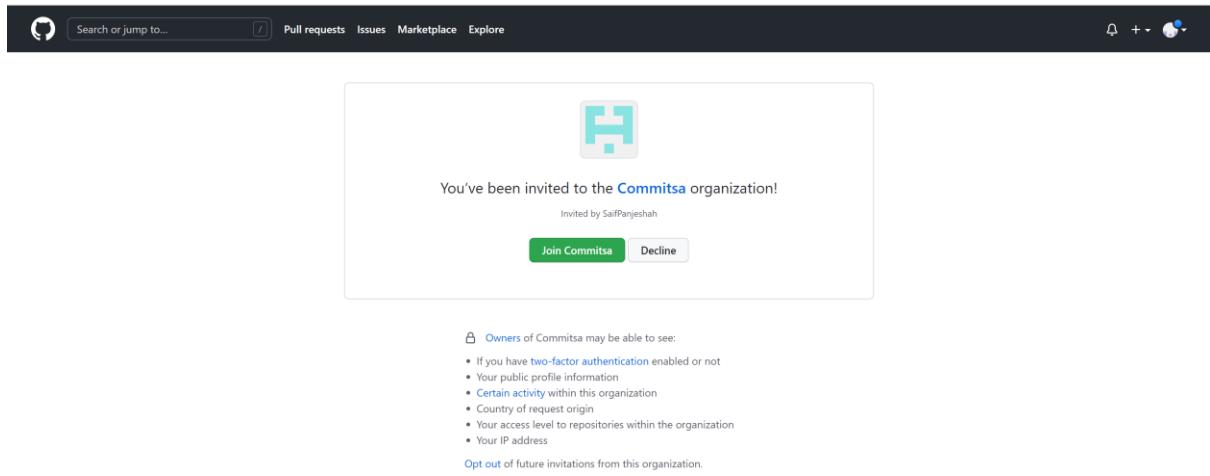


Fig. Inviting member to Commitsa organization

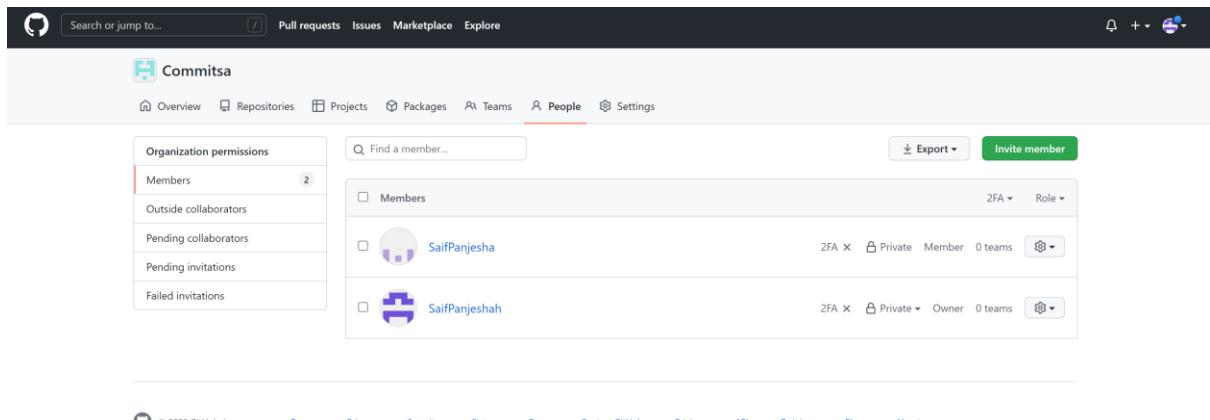
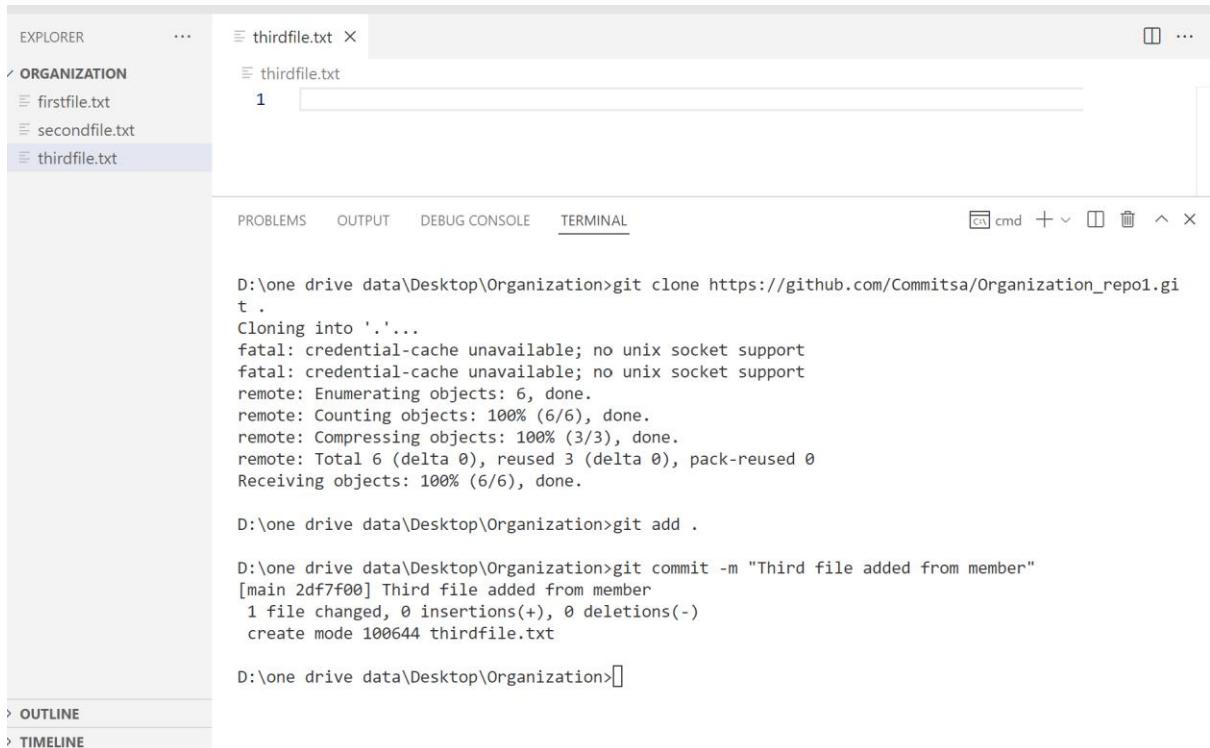


Fig. Successful Invitation

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 active. The terminal output is as follows:

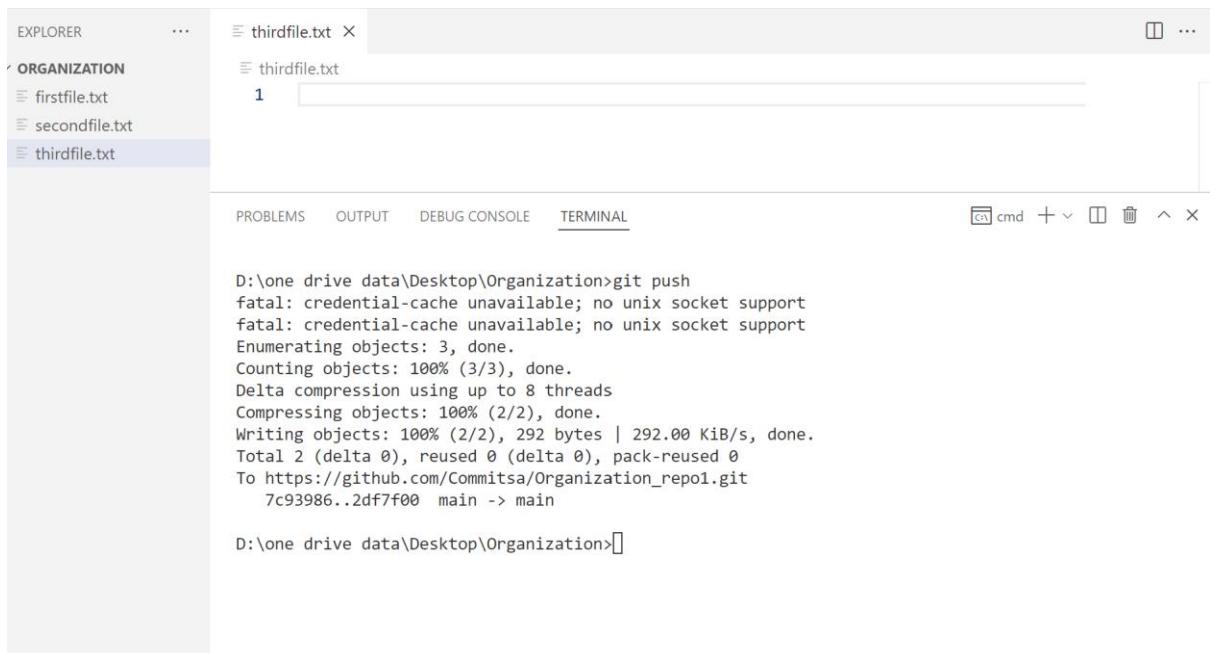
```
D:\one drive data\Desktop\Organization>git clone https://github.com/Commitsa/Organization_repo1.git .
Cloning into '.'...
fatal: credential-cache unavailable; no unix socket support
fatal: credential-cache unavailable; no unix socket support
remote: Enumerating objects: 6, done.
remote: Counting 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 active. The terminal output 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

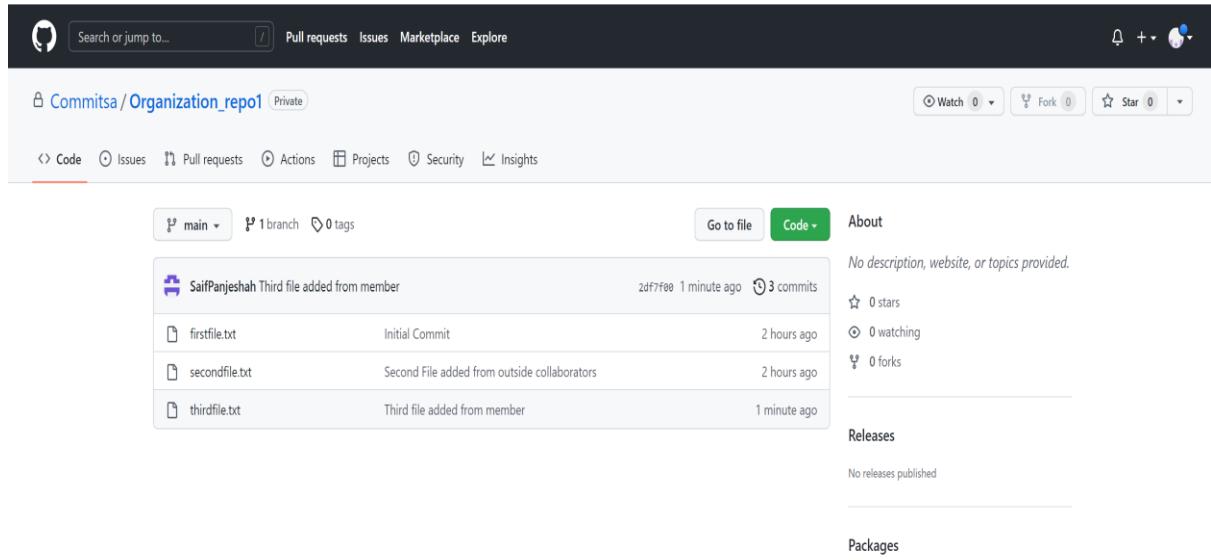
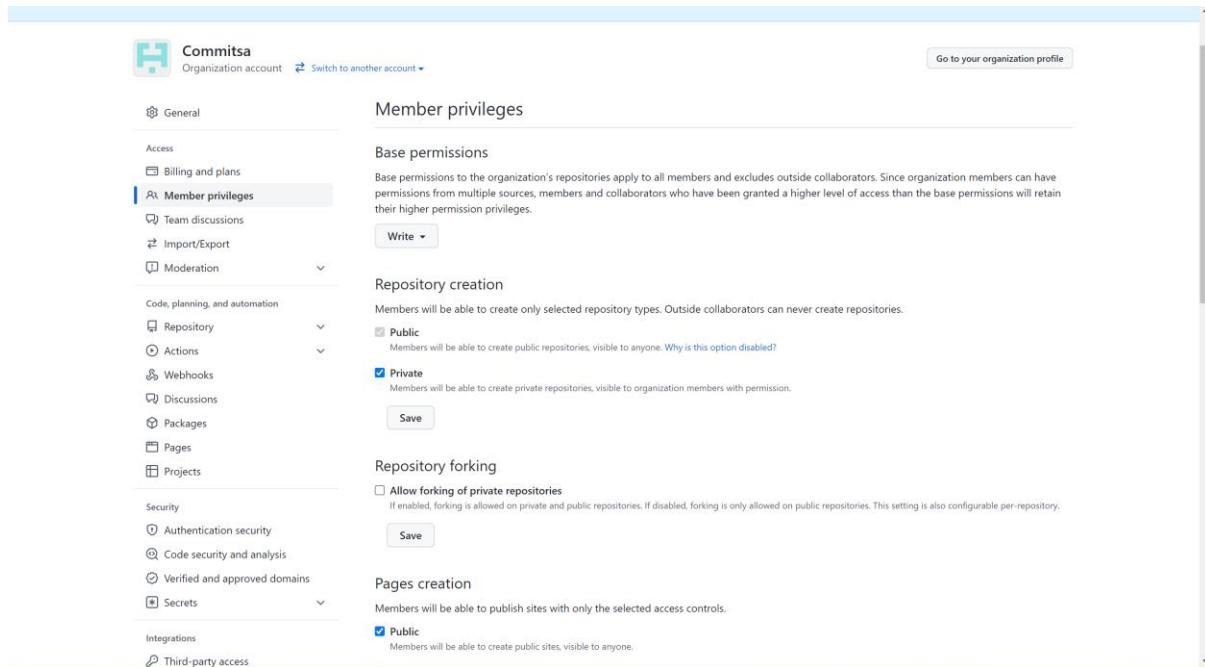


Fig. Successful Push on GitHub

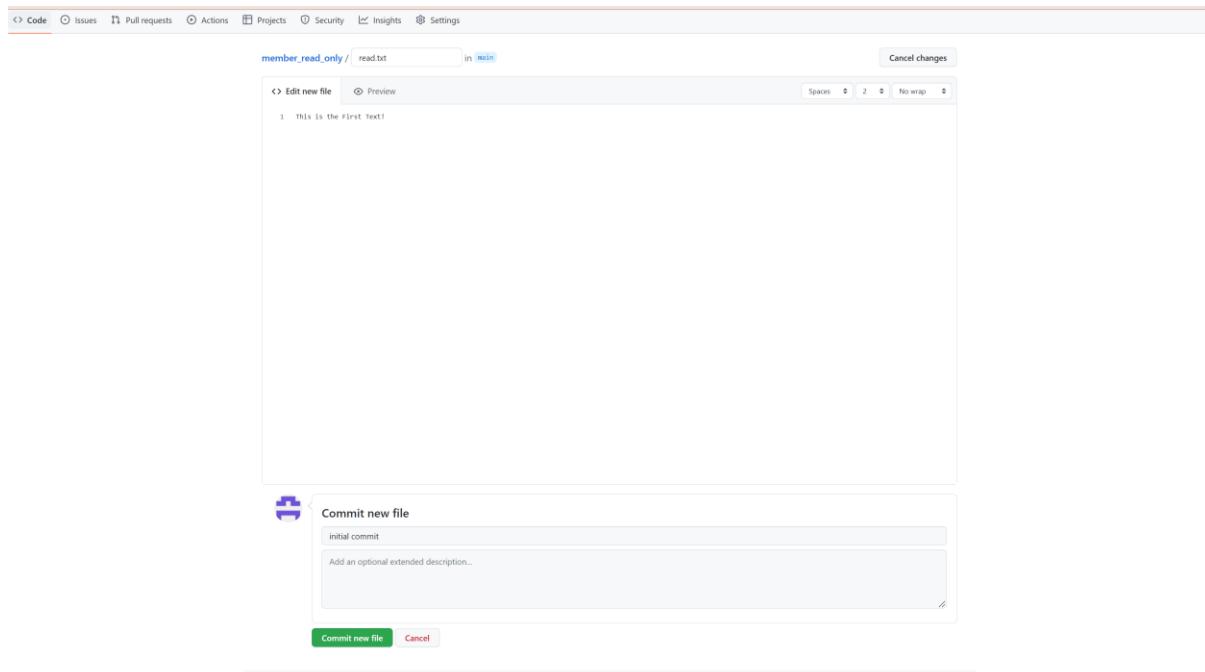
Failing to Manage Access for Individual Repositories:



The screenshot shows the 'Member privileges' section of the Commitsa organization account settings. The left sidebar has 'Member privileges' selected under 'Access'. The main area shows 'Base permissions' set to 'Write'. Under 'Repository creation', the 'Public' checkbox is unchecked, and the 'Private' checkbox is checked. Under 'Repository forking', the 'Allow forking of private repositories' checkbox is unchecked. Under 'Pages creation', the 'Public' checkbox is checked. A 'Save' button is visible at the bottom of each section.

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'. A file named 'read.txt' contains the text 'This is the first text!'. Below the file list, a 'Commit new file' dialog is open, showing an 'initial commit' message and a text input field. Buttons for 'Commit new file' and 'Cancel' are at the bottom of the dialog.

Fig. Creating member read only repositories

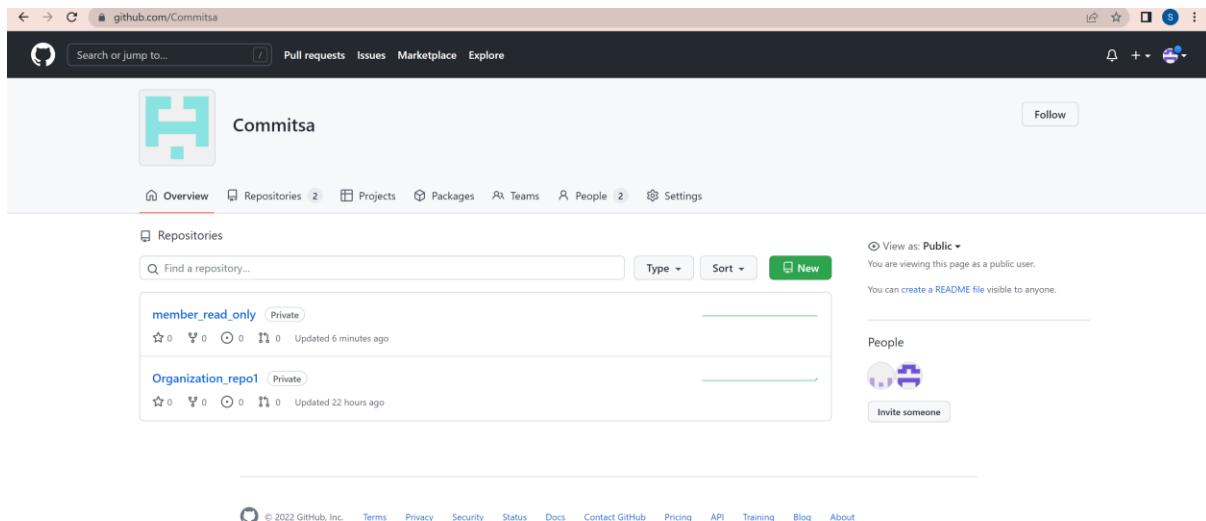


Fig. Successful Created Account

Checking Access for Individual Repositories:

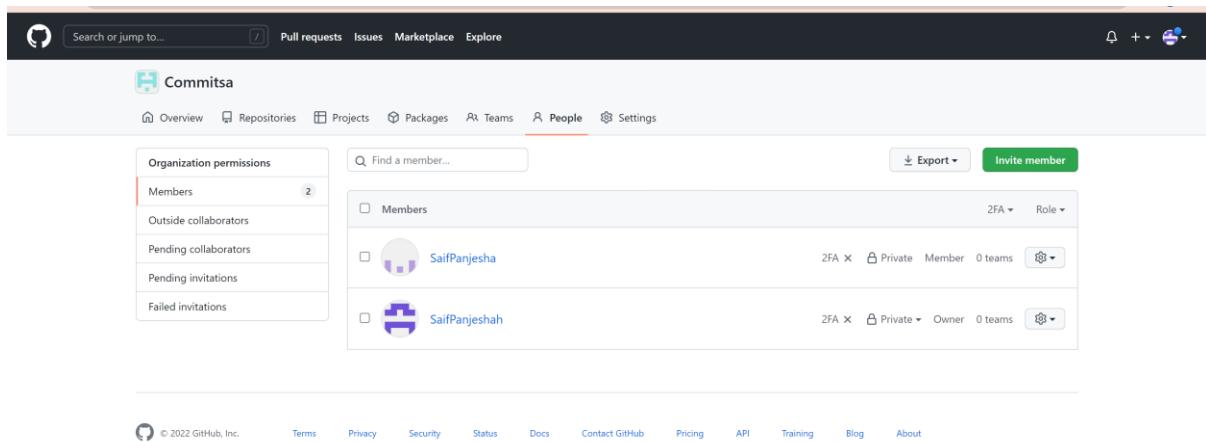


Fig. Checking access

The screenshot shows the GitHub organization profile for 'Commitsa'. On the left, there's a sidebar with the user's profile picture and name, 'SaifPanjesha'. Below it, it says 'Role: Member' and lists '2 repositories', '0 teams', and 'Membership private'. There are buttons for 'Convert to outside collaborator' and 'Remove from organization'. The main area shows a list of repositories: 'Commitsa/Organization_repo1' with 'Write on this repository' and 'Manage access' buttons, and 'Commitsa/member_read_only' with similar buttons. A search bar at the top right says 'Find a repository they have access to...'. At the bottom, there's a footer with links like Terms, Privacy, Security, Status, Docs, Contact GitHub, Pricing, API, Training, Blog, and About.

Fig. Shows both the account has same access for repositories

The screenshot shows the GitHub organization settings for 'Commitsa'. The left sidebar has sections for General, Access (selected), Member privileges, Code, planning, and automation, Security, and Secrets. Under 'Access', 'Member privileges' is selected. A modal dialog 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. The 'Change base permission to "Read"' button is highlighted. The background shows the organization's member permissions: 'No permission' (disabled), 'Read' (selected), and 'Write'. The 'Write' option is described as allowing members to clone, pull, and push all repositories. Other sections visible include 'Organization member permissions' (with 'No permission' selected) and 'Pages creation' (with 'Read' selected).

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 links for "Pull requests", "Issues", "Marketplace", and "Explore". Below the navigation is a search bar and a user profile icon. The main content area features a title "Seamless communication with teams" and a subtitle "Teams are a great way for groups of people to communicate and work on code together. Take a look at why they're great." Three circular icons represent different team features: "Flexible repository access" (repositories), "Request to join teams" (people), and "Team mentions" (mentions). Below each icon is a brief description and a "New team" button.

Fig. Teams Ribbons of Organizational Account

The screenshot shows the "Create new team" form on GitHub. The header "Create new team" is visible. The "Team name" field contains "Commitsa_Read_Write_teams" with a green checkmark. A note below it says "Mention this team in conversations as @Commitsa/commitsa_read_write_teams.". The "Description" field is empty. The "Parent team" section indicates "There are no teams that can be selected.". The "Team visibility" section has two options: "Visible" (selected) and "Secret". A note for "Visible" says "A visible team can be seen and @mentioned by every member of this organization." A note for "Secret" says "A secret team can only be seen by its members and may not be nested." At the bottom is a "Create team" button.

Fig. Creating Teams in Organization

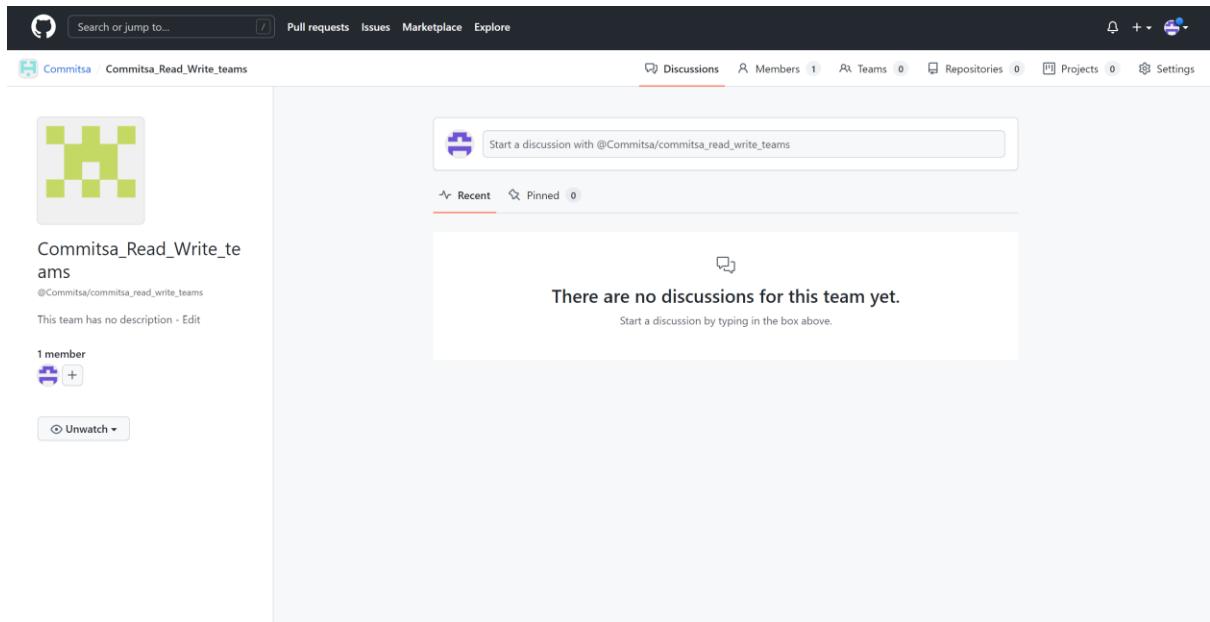


Fig. Read and Write team successfully Created

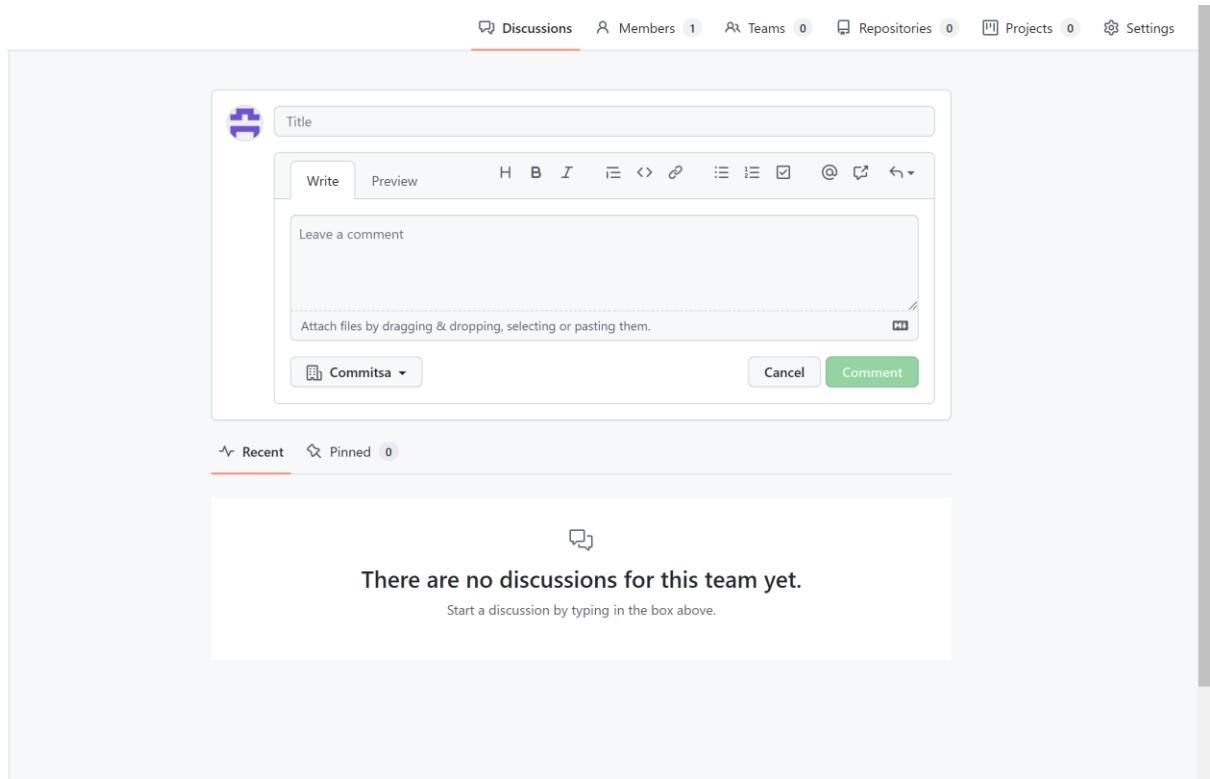


Fig. Forum of discussion ribbons in teams to check

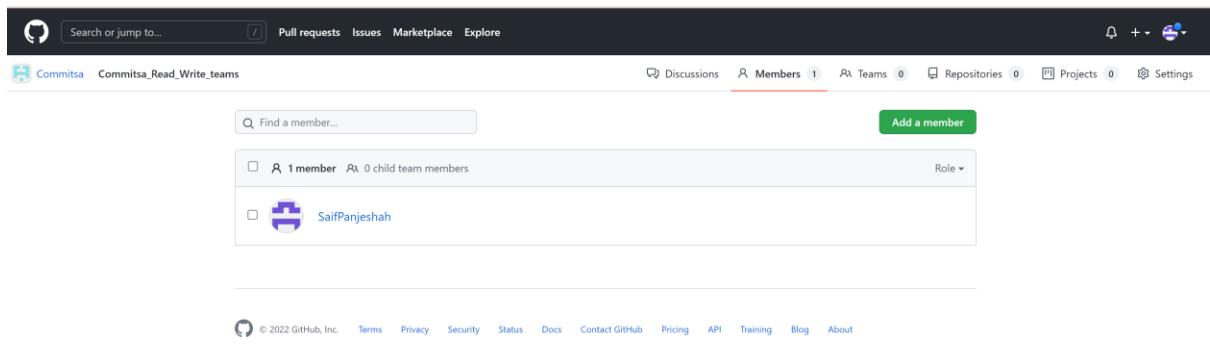


Fig. Owner as Team Member Default

Let's create team repository:

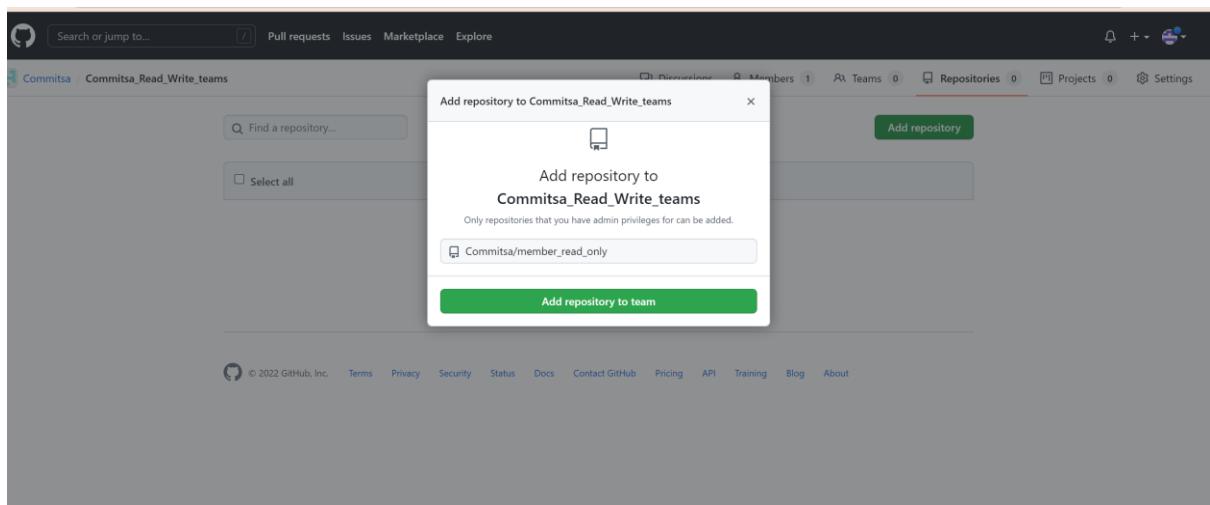


Fig. Creating Team Repository

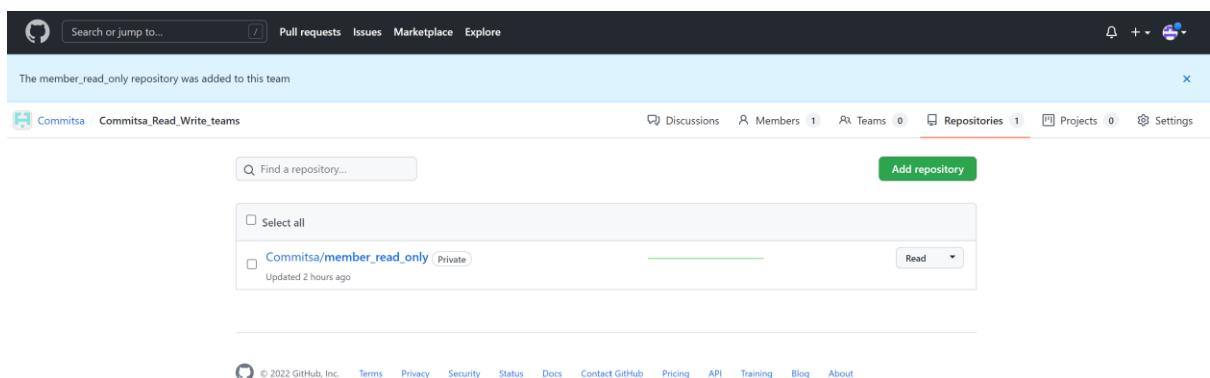


Fig. Repository Created with read access

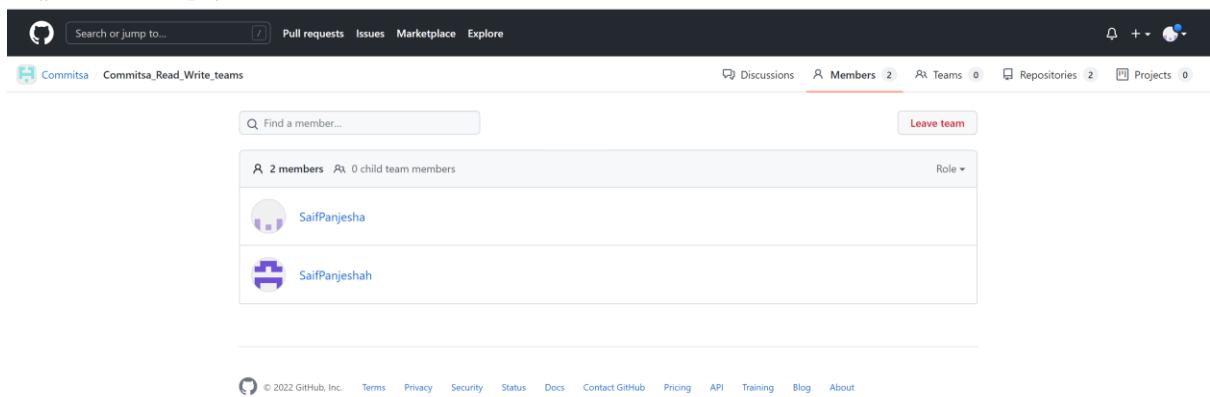


Fig. Successful adding team member

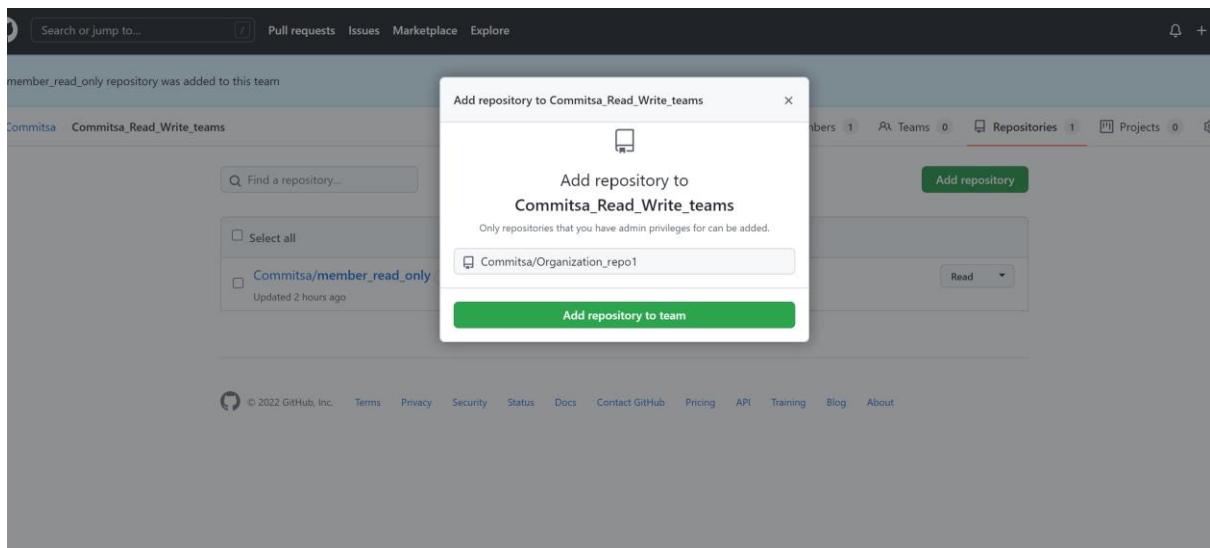


Fig. Creating Team Repository

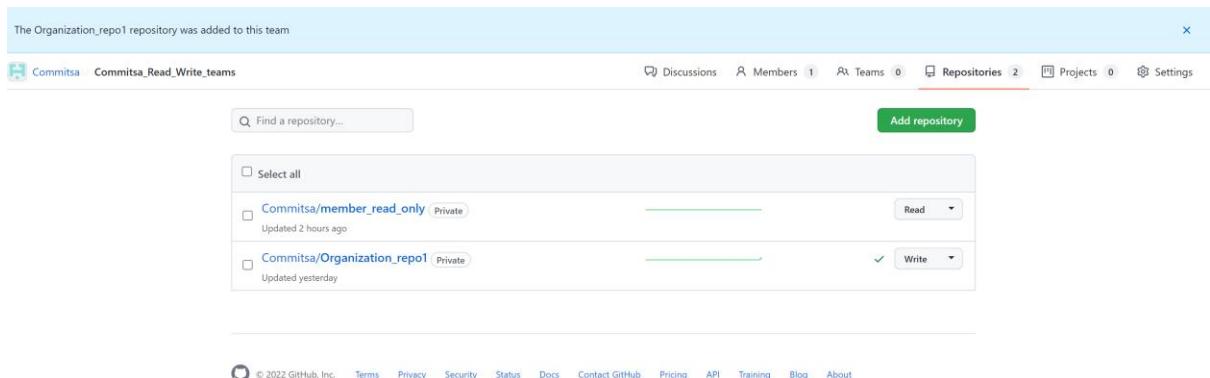
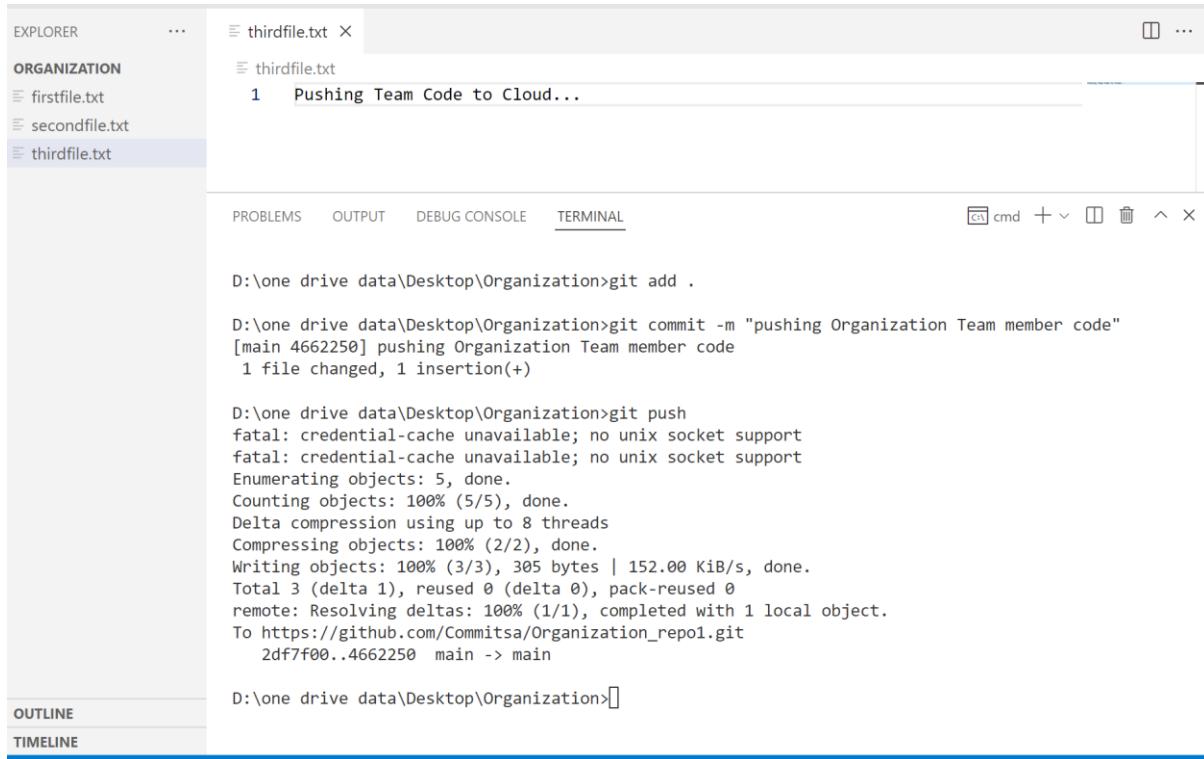


Fig. Successful Created repositories with write access

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. In the Explorer sidebar, there are three files: firstfile.txt, secondfile.txt, and thirdfile.txt. The terminal tab is active, displaying 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/Commitisa/Organization_repo1.git  
 2df7f00..4662250 main -> main  
  
D:\one drive data\Desktop\Organization>
```

Fig. Successful Push coding on Organization repository with write access

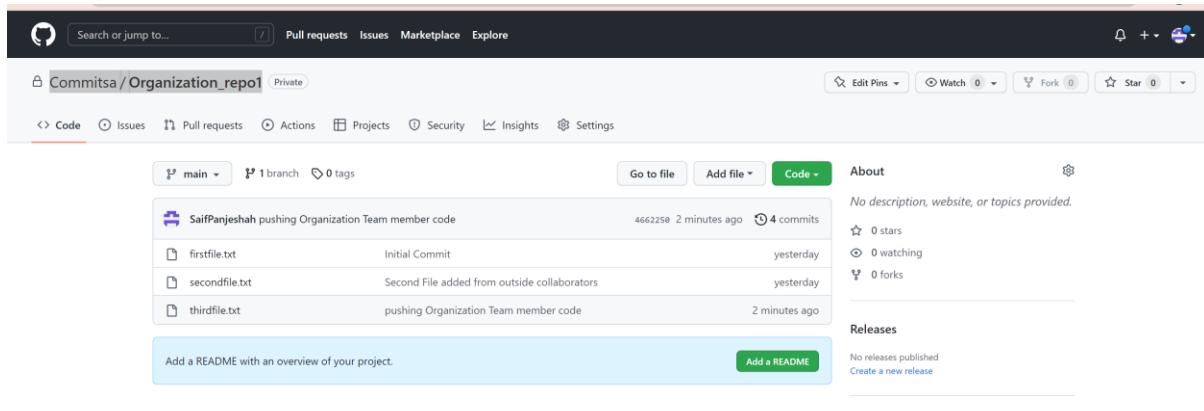


Fig. Successful Push on GitHub

Let's Push our code on: https://github.com/Commitsa/member_read_only.git.

The screenshot shows the Visual Studio Code interface. In the Explorer sidebar, there is a folder named 'Organizations' containing a file named 'read.txt'. The content of 'read.txt' is:

```
1 This is the First Text!
2 Let's Push Code on Cloud !!!
```

In the Terminal tab, the user has run several Git commands:

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

D:\one drive data\Desktop\Organizations>git clone https://github.com/Commitsa/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 "Successful Push Code on Cloud .."
[main 696f26a] Successful 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/Commitsa/member_read_only.git
  f5d1833..696f26a main -> main

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

Fig. Successful Push coding on member_read_only repository with read access

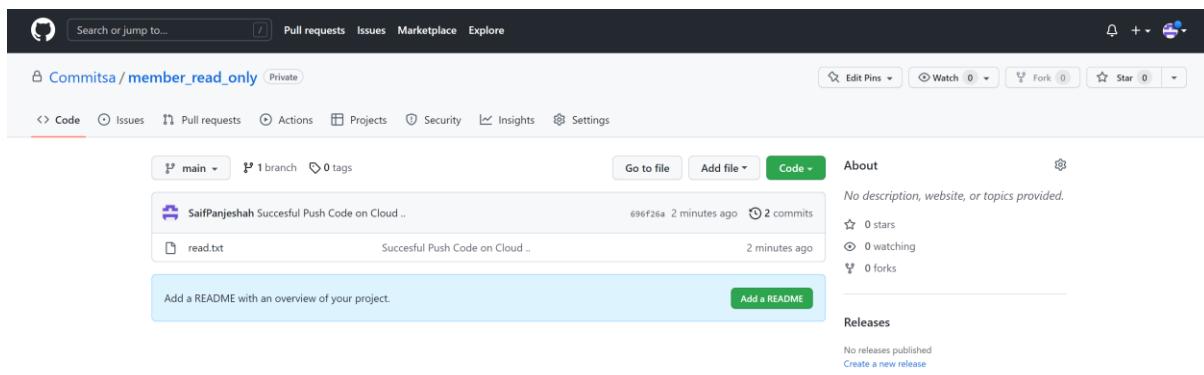


Fig. Successful Push on GitHub

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 ' git clone ' 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

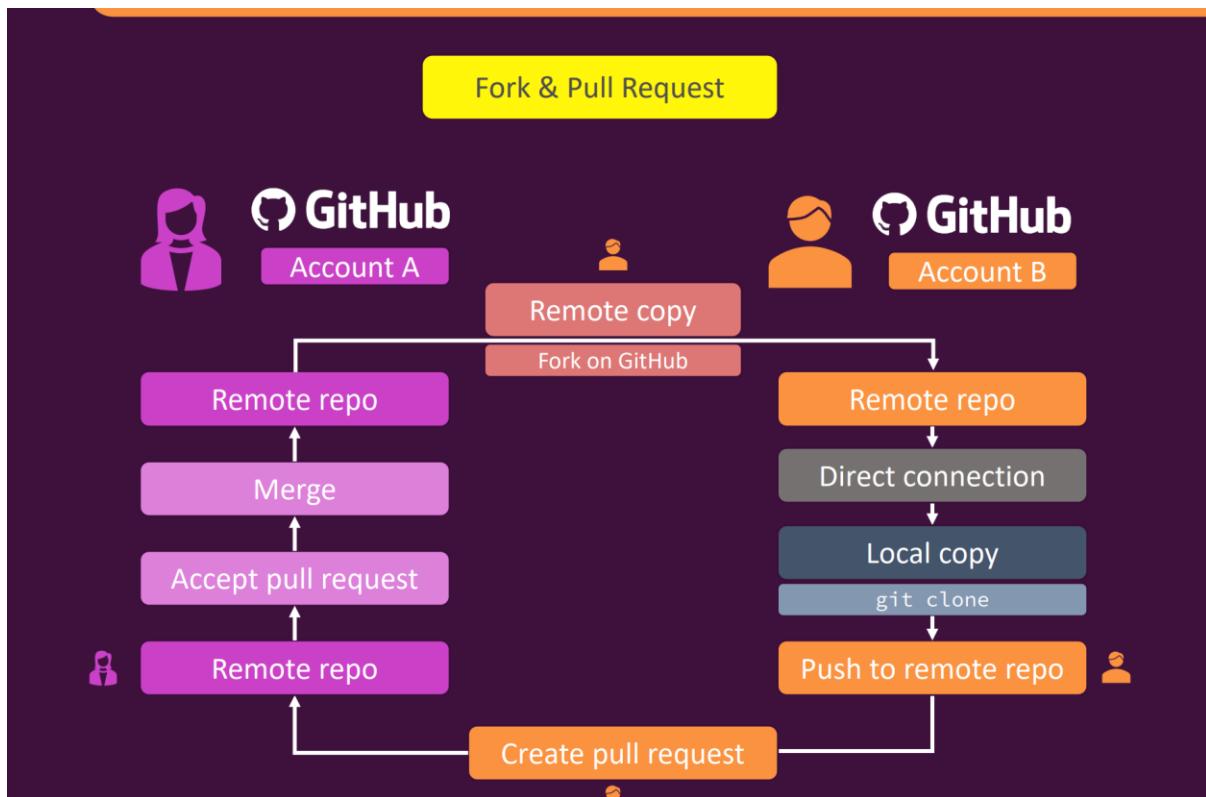


Fig. Creating Fork & Pull Request

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

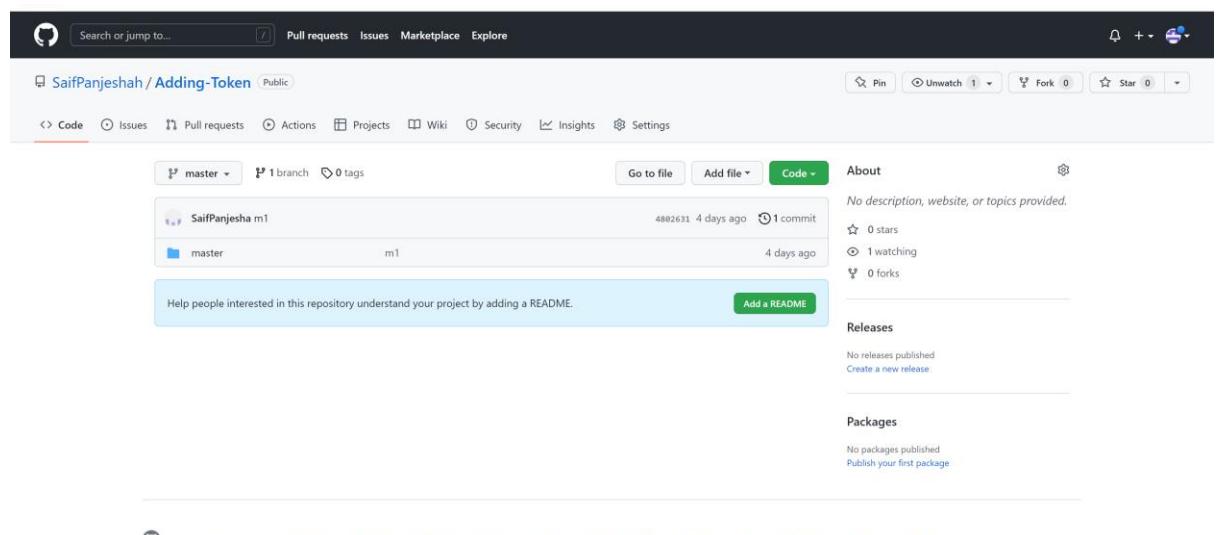


Fig. Forking the repository from another account

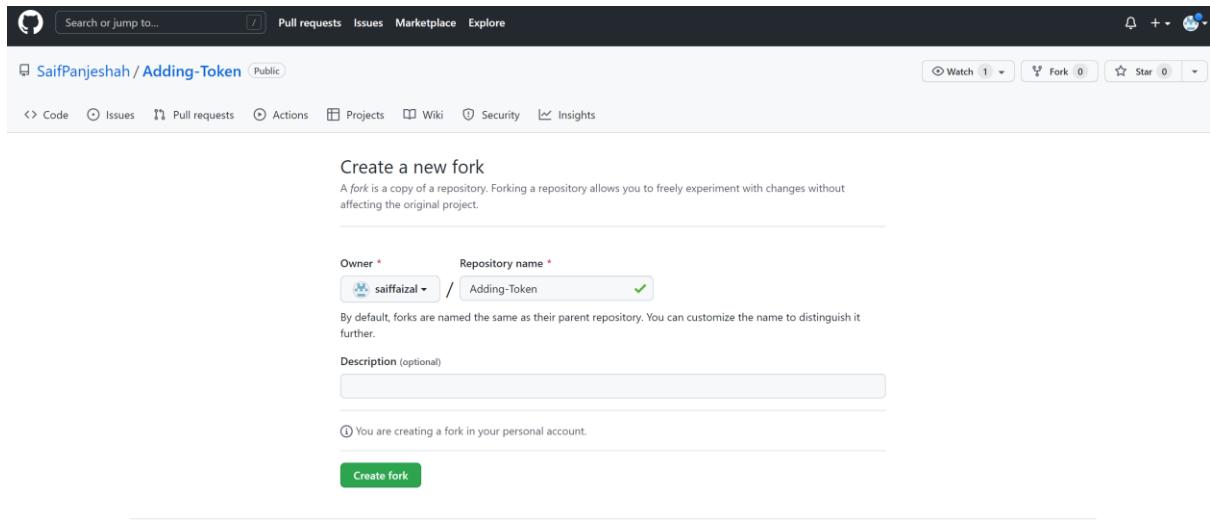


Fig. Create a new fork

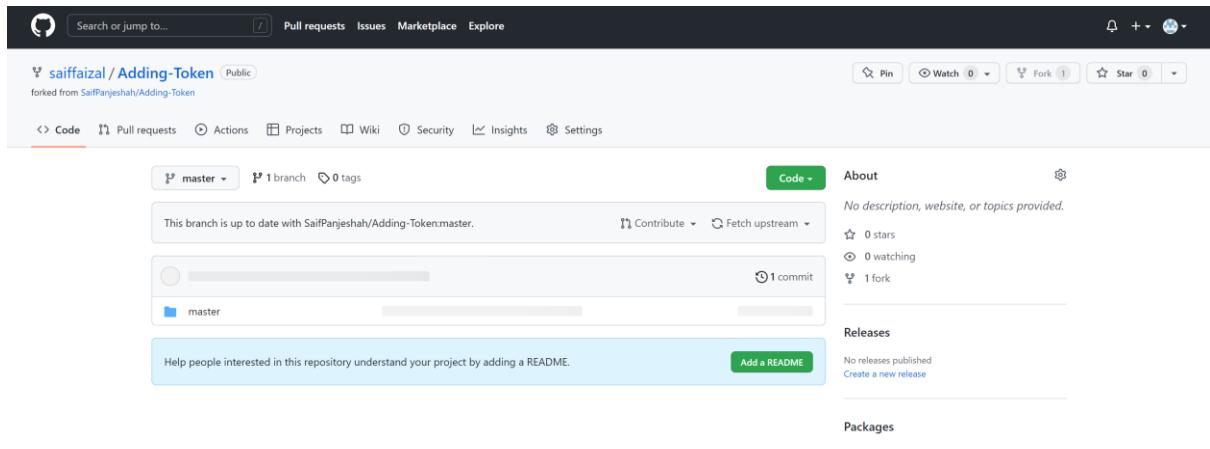


Fig. Forked Successful

2. Clone your new fork locally

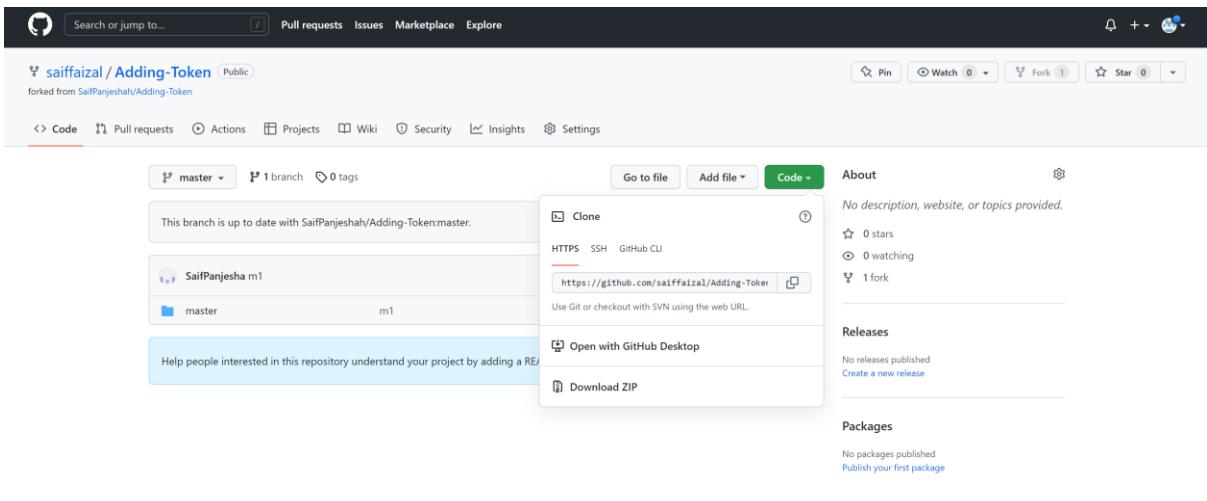


Fig. cloning the repository.

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.

The screenshot shows a Visual Studio Code interface with a terminal window open. The terminal output is as follows:

```
(c) Microsoft Corporation. All rights reserved.

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

D:\one drive data\Desktop\Forking Repositories>
```

Fig. Success to push the code

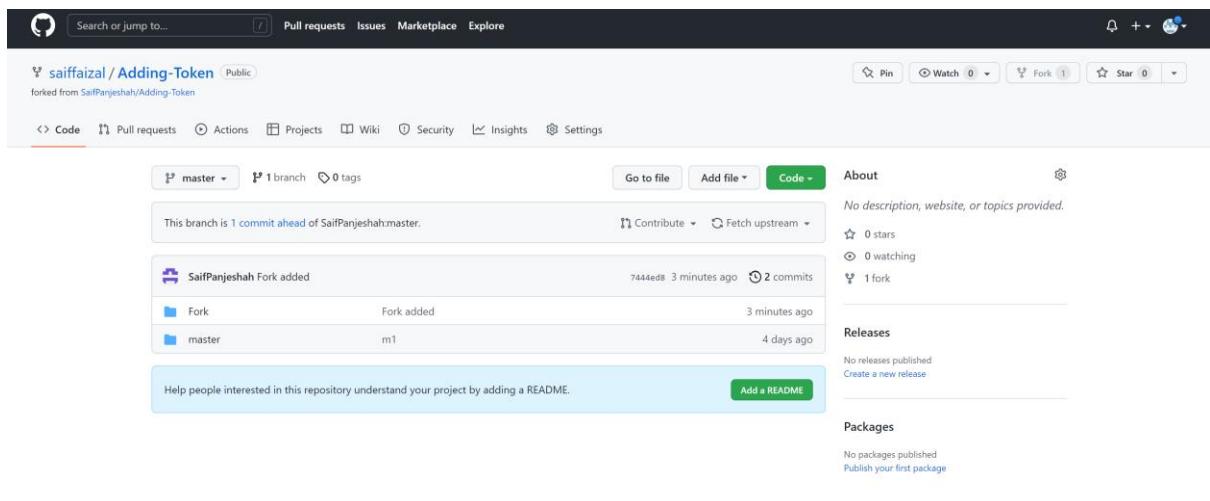


Fig. Successful on GitHub to member account

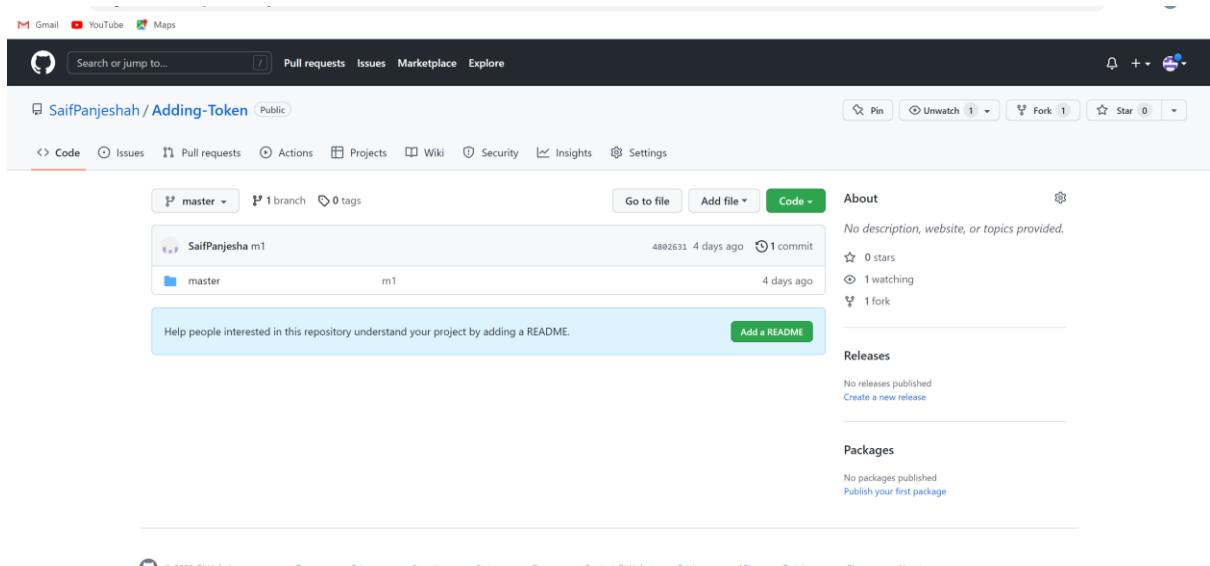


Fig. Not Fork Update Successful on GitHub owner account

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

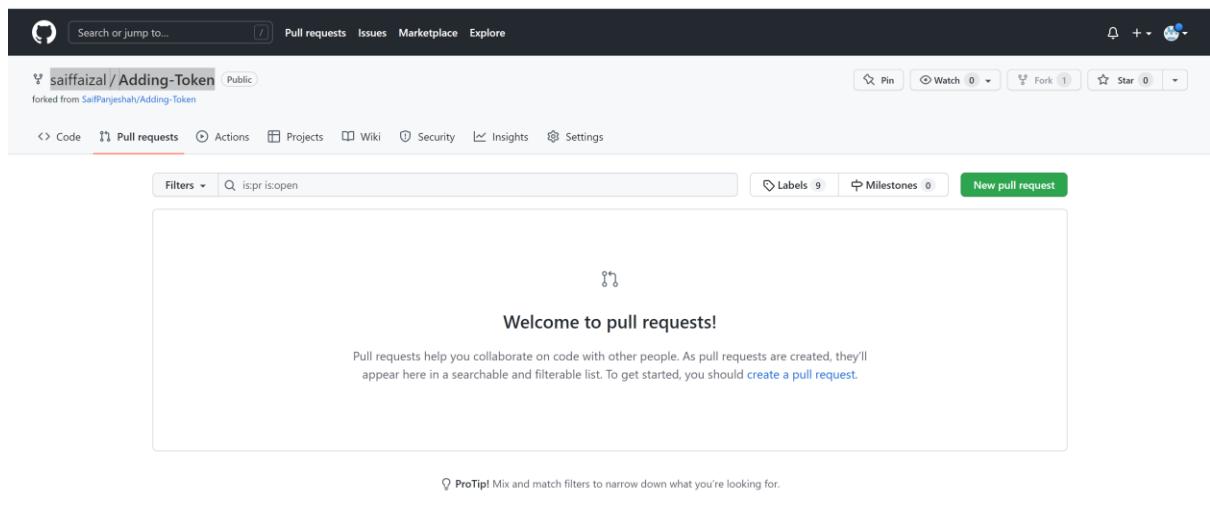


Fig. Creating a pull request

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.

The screenshot shows a GitHub comparison interface. At the top, it displays the repositories: 'base repository: SaifPanjeshah/Adding-Token' and 'head repository: saiffaizal/Adding-Token'. The 'base' dropdown is set to 'master', and the 'head' dropdown is also set to 'master'. A green checkmark indicates that the branches are 'Able to merge'. Below this, there's a section titled 'Comparing changes' with a note: 'Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks.' A 'Create pull request' button is visible. The main area shows a single commit from 'SaifPanjeshah' on May 26, 2022, which added a file named 'Fork/addingnewfeature2.txt'. The commit message is 'Fork added'. The commit details show an empty file.

Fig. Comparing Changes

4. Type a title and description for your pull request.

The screenshot shows the GitHub pull request creation interface. It starts with a header 'Open a pull request' and a note: 'Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.' The base repository is 'SaifPanjeshah/Adding-Token' and the head repository is 'saiffaizal/Adding-Token'. The 'base' dropdown is set to 'master' and the 'head' dropdown is set to 'master'. A green checkmark indicates that the branches are 'Able to merge'. The main area is titled 'Adding a Fork Design' and contains a rich text editor with the placeholder text 'Designed with Complete Code on adding new features'. There's a checkbox for 'Allow edits by maintainers'. A 'Create pull request' button is at the bottom right. Below the editor, it shows a single commit from 'SaifPanjeshah' on May 26, 2022, which added a file named 'Fork/addingnewfeature2.txt'. The commit message is 'Fork added'. The commit details show an empty file.

Fig. open pull request

5. To create a pull request that is ready for review, click Create Pull Request.

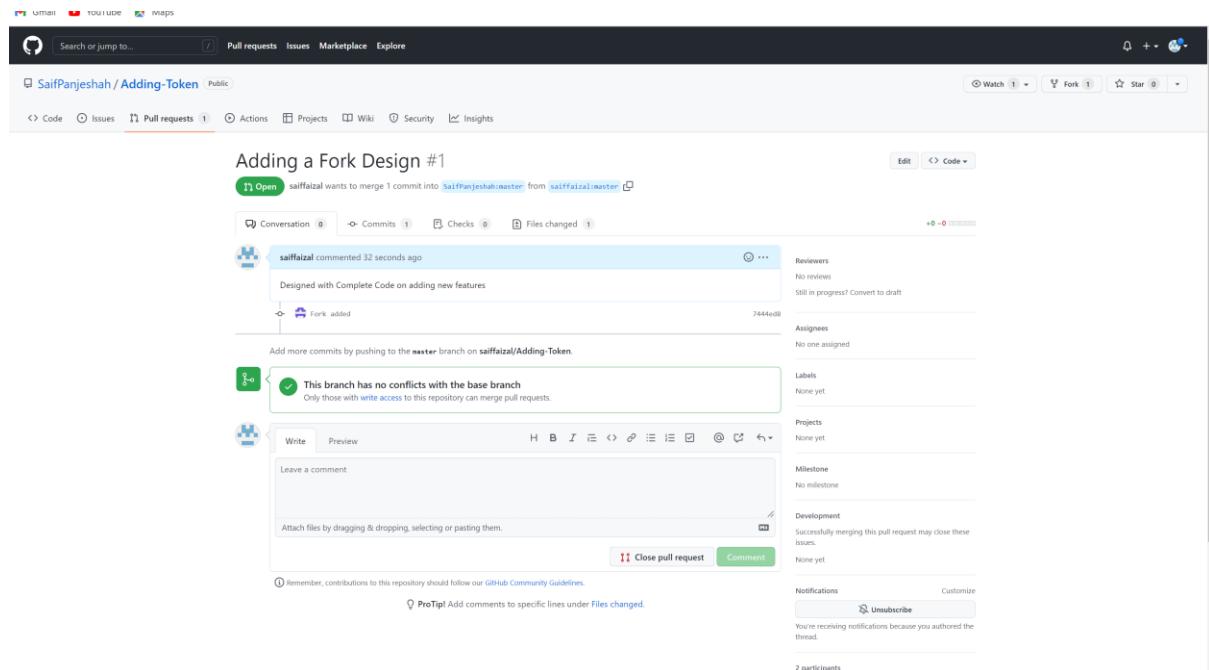


Fig. open pull request successful

Now, to view this pull request open owner organizational account

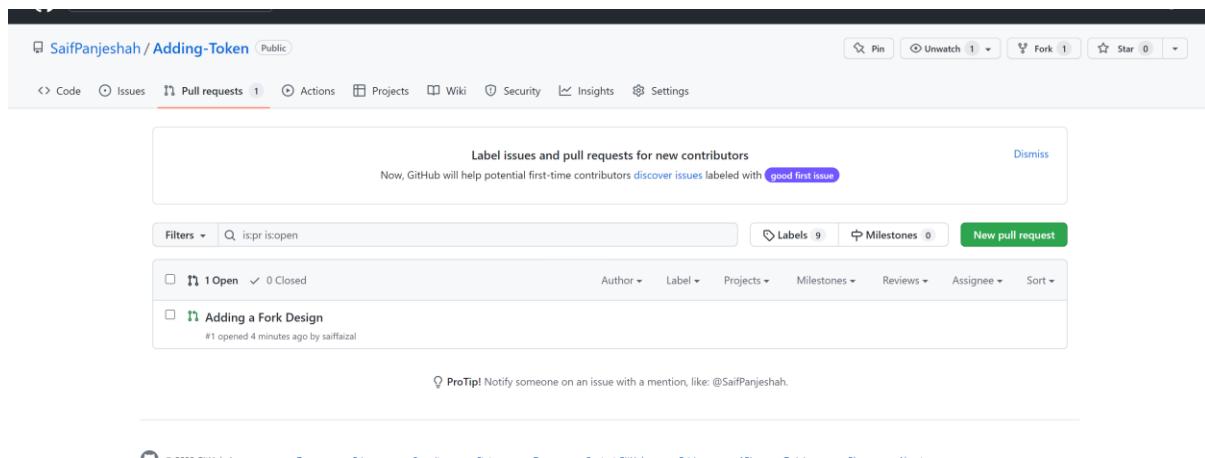


Fig. Owner view the pull request

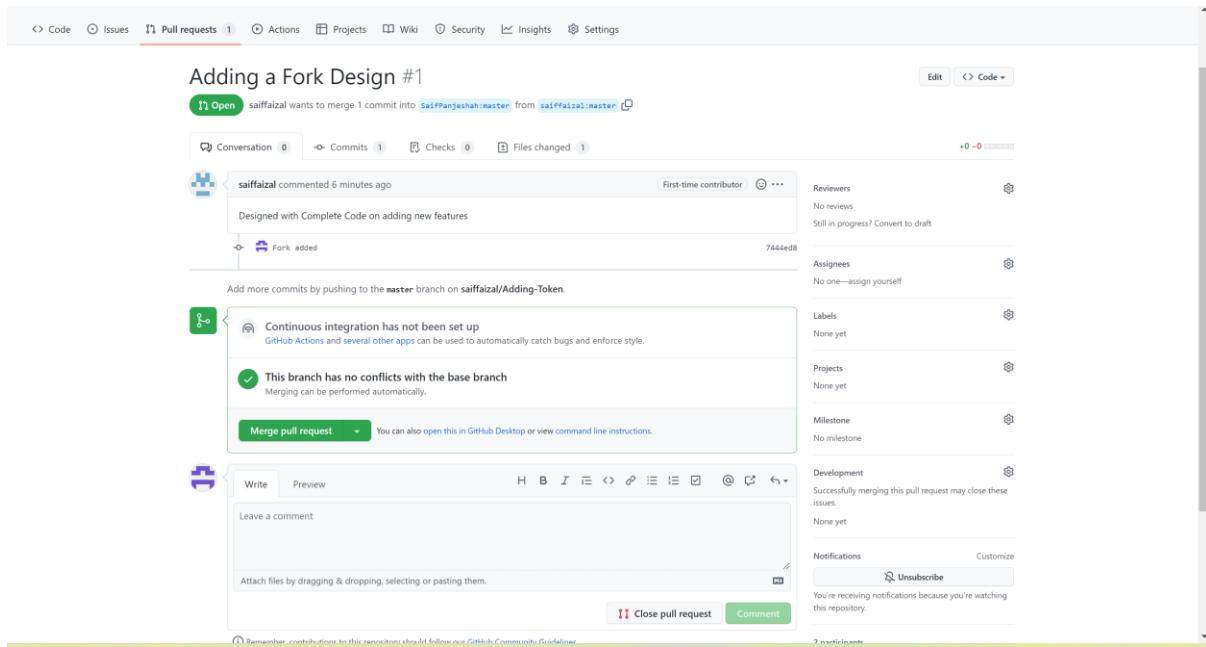


Fig. Merging or closing pull request

If owner don't like and don't want to merge this pull request simply close this pull request

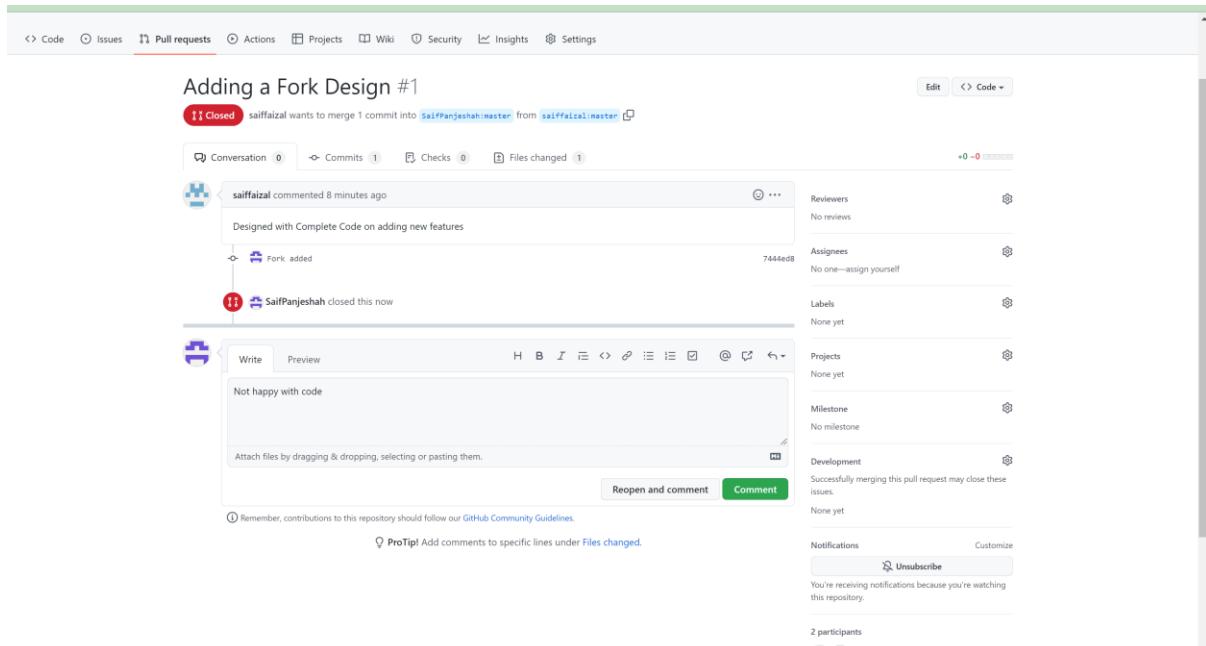


Fig. Owner closing this pull request

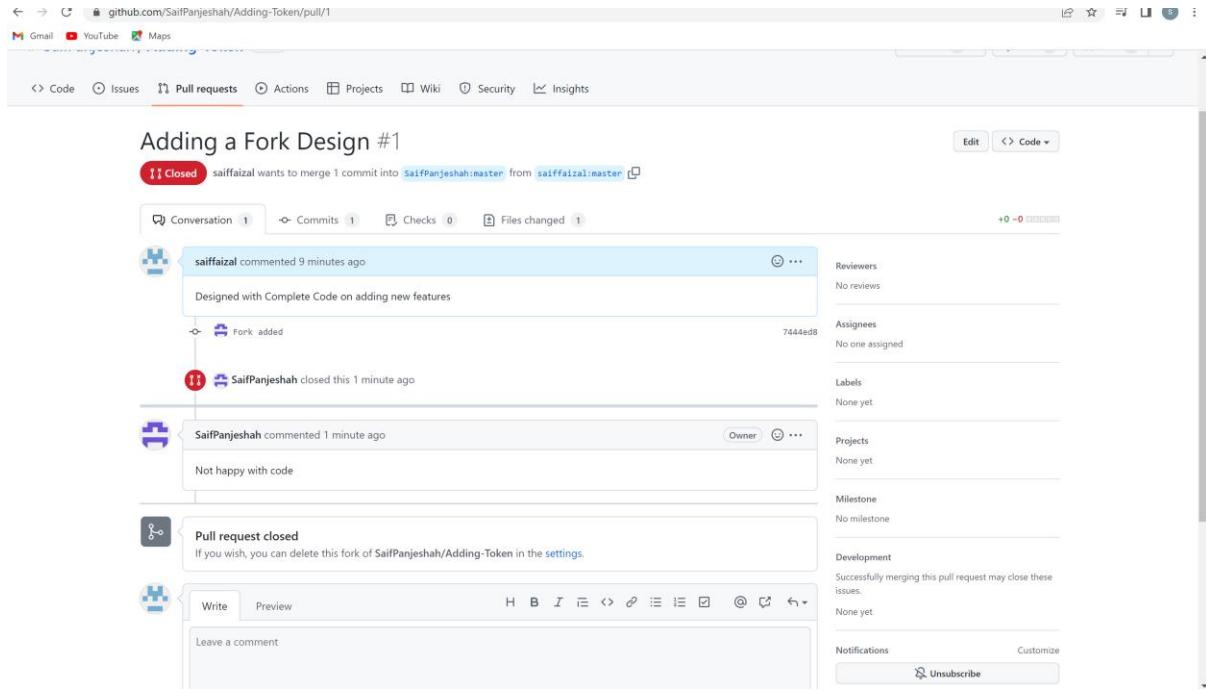


Fig. Closed Successful view from Member account

Now, Owner Thinks a Second Way like the Fork design and wants to merge the code

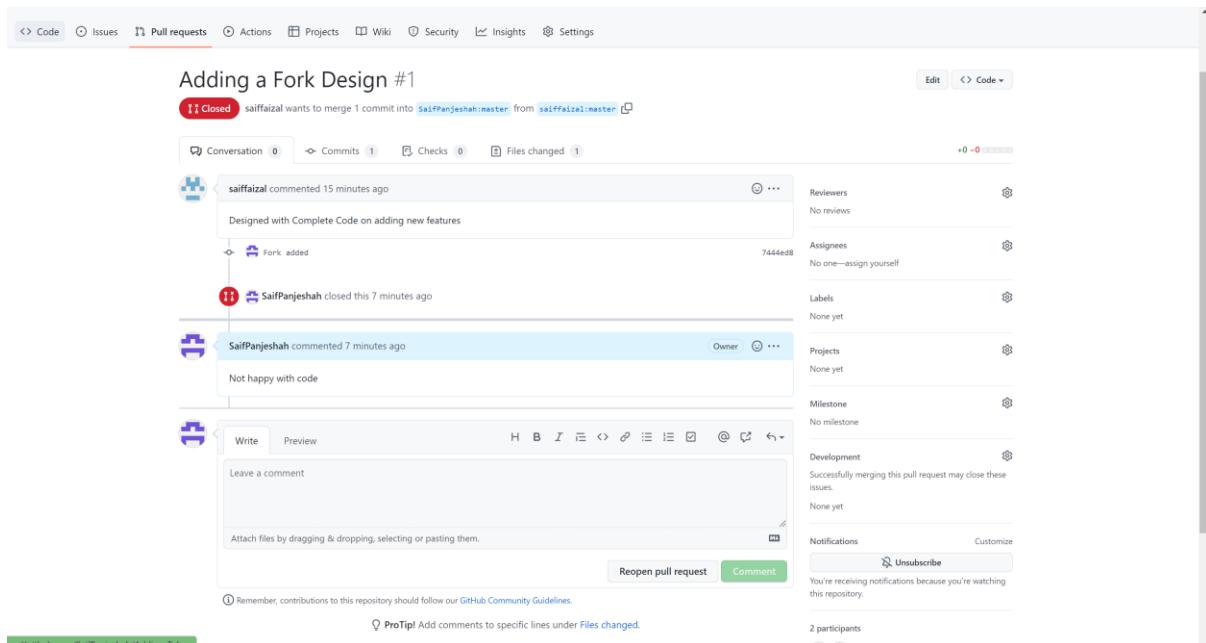


Fig. reopen the pull requests

Merging the pull requests:

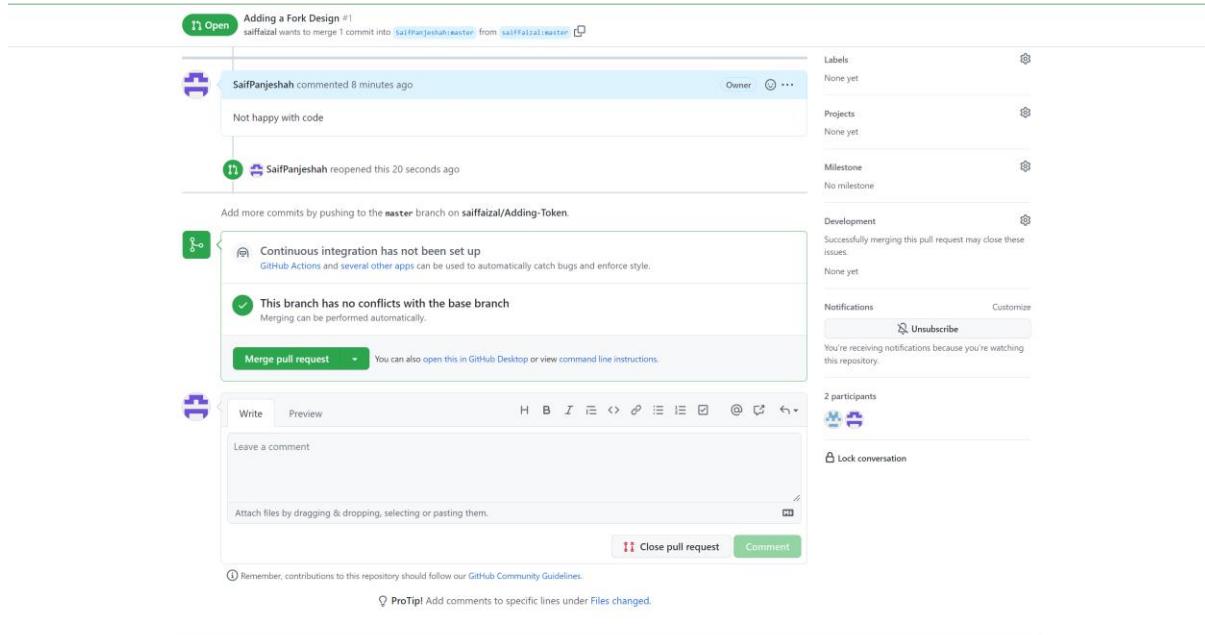


Fig. Merging the pull requests

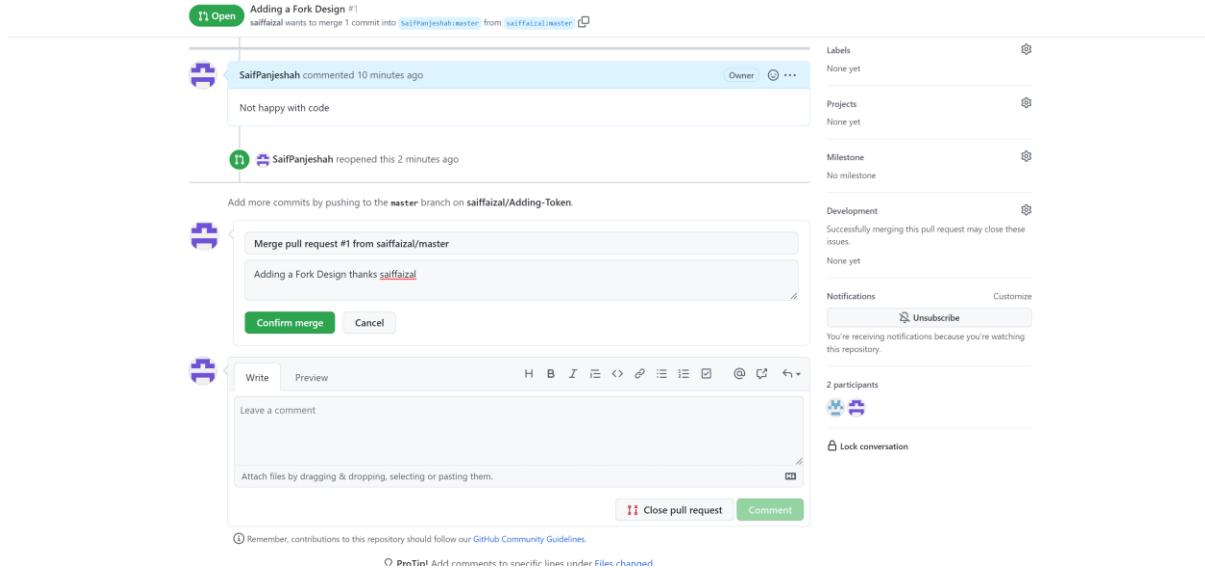


Fig. Confirm merging

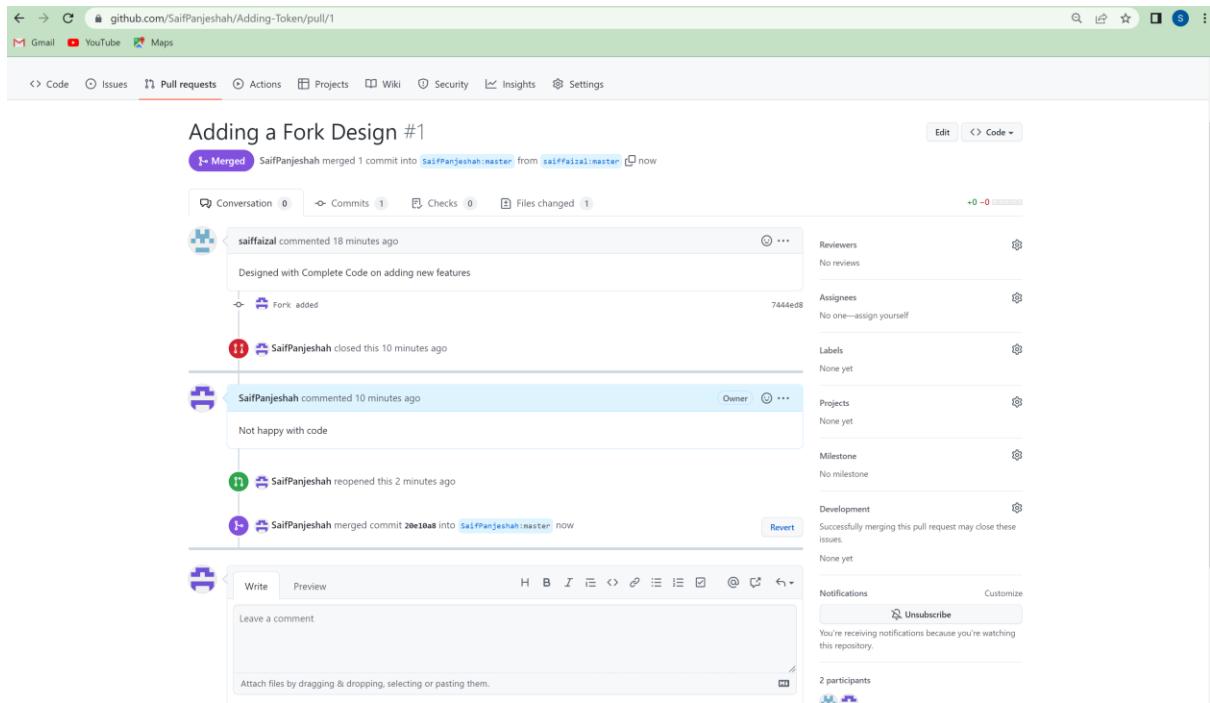


Fig. Merge successful

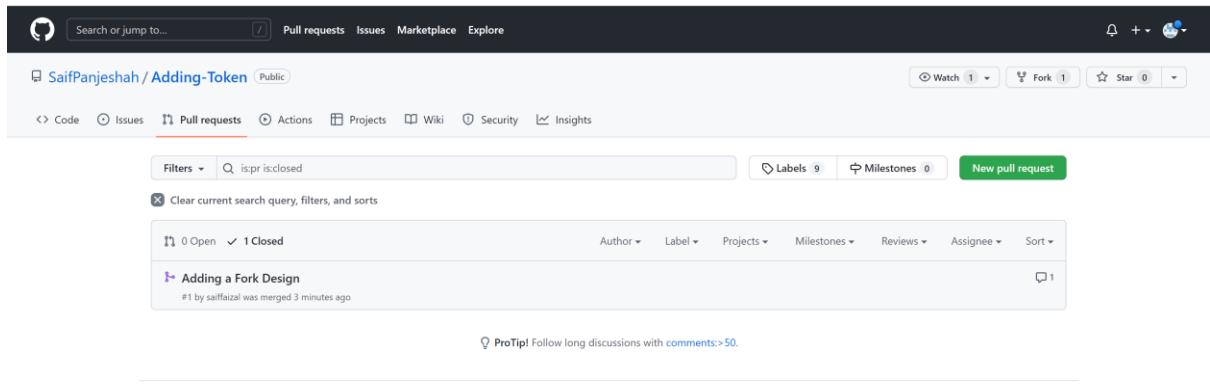


Fig. Close and successful Merge view from member account

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.

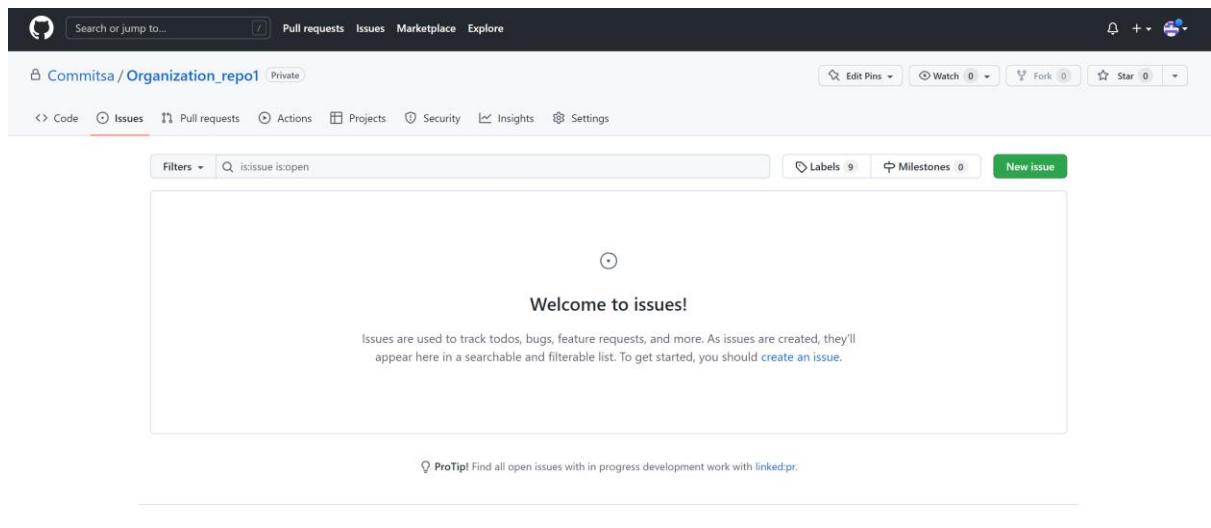


Fig. Creating a new Issue on organizational account repository

2. Click New issue:

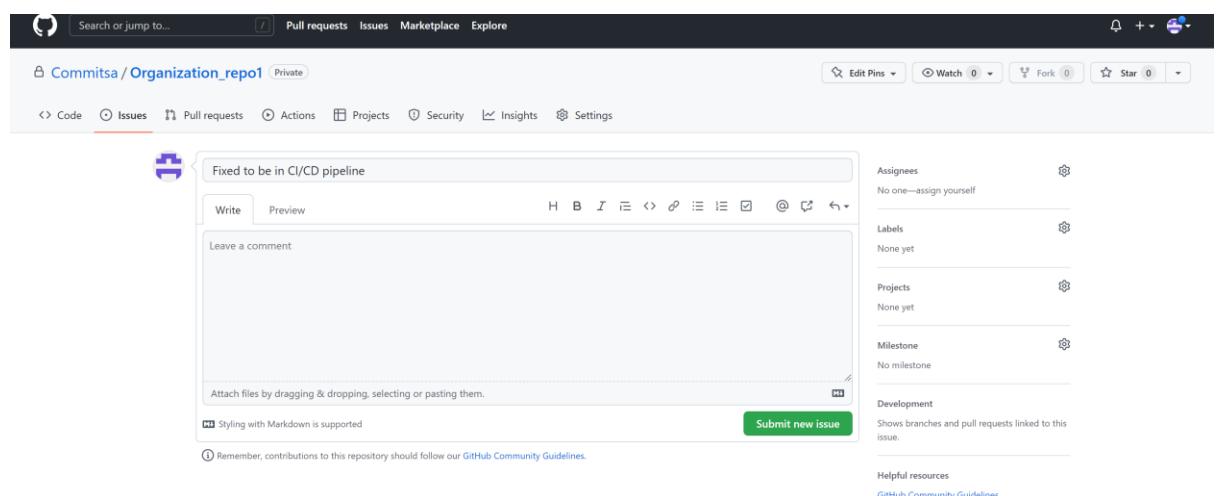


Fig. Creating Issues with CI/CD Pipelines of Devops

Lets assigned this to member user of this organization and add labels as bugs needed to be fix:

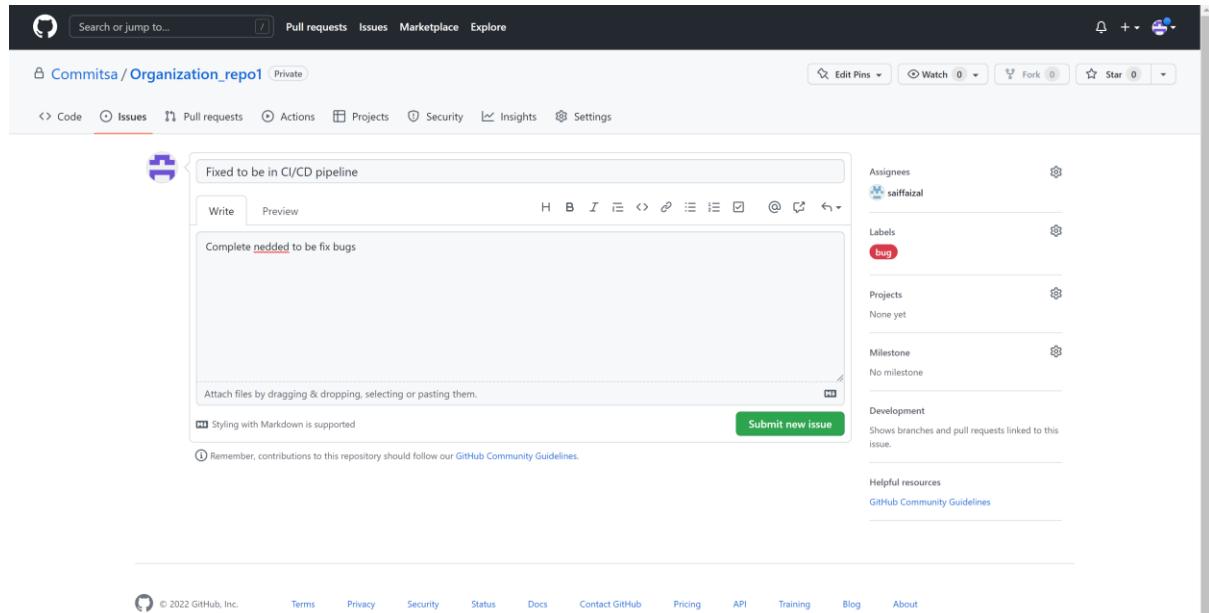


Fig. Fixed needs to be done

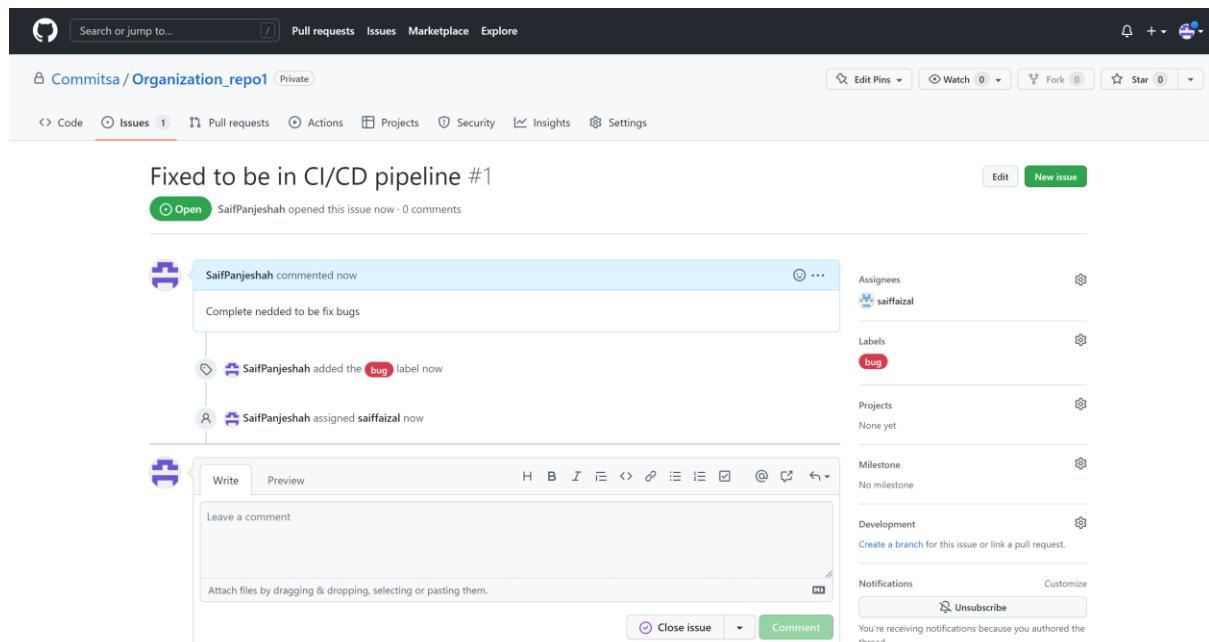


Fig. Issue successful Created

Closing the Issue:

The screenshot shows the GitHub Issues page for the repository 'Commitsa / Organization_repo1'. The page displays one open issue, which is a bug titled 'Fixed to be in CI/CD pipeline'. The issue was opened by SaifPanjeshah 2 minutes ago. The GitHub interface includes standard navigation bars like Code, Issues, Pull requests, Actions, Projects, Security, and Insights. It also features filters, labels (9), milestones (0), and a 'New issue' button.

Fig. view on member account

This screenshot shows the detailed view of the issue '#1 Fixed to be in CI/CD pipeline'. The issue is marked as 'Open' and was created by SaifPanjeshah 3 minutes ago. The issue has 0 comments. The issue details include a comment from SaifPanjeshah stating 'Complete nedded to be fix bugs', and another comment adding the 'bug' label and assigning it to 'saiffaizal'. The issue is currently unassigned and has no milestones or projects assigned. The GitHub interface shows the 'Write' and 'Preview' tabs for commenting, along with various rich text and file upload options. Notifications and customization options are also visible on the right side.

Fig. Member Fixed this issue

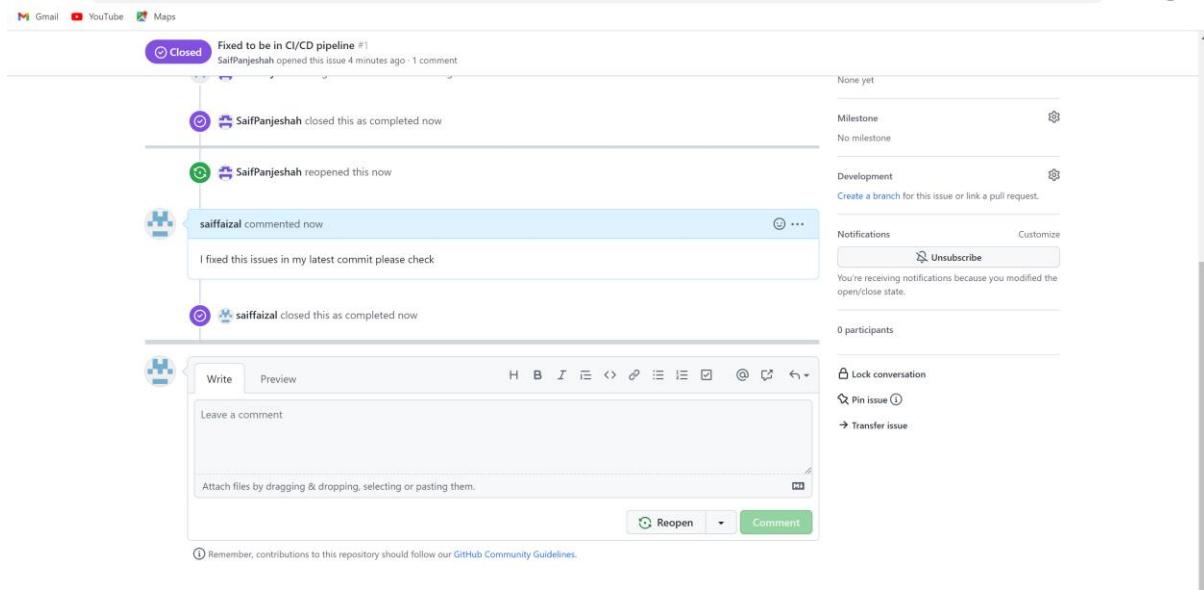


Fig. successful added close and fix commits

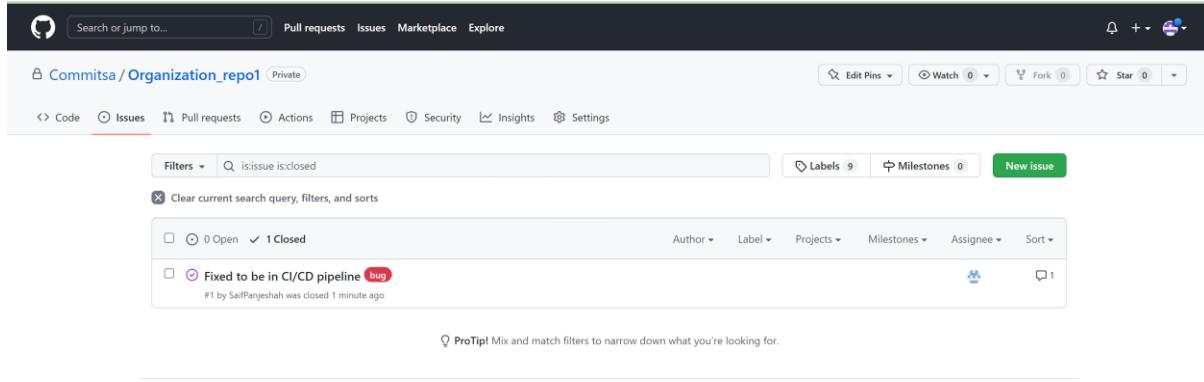


Fig. Successful close this Issue

Working with GitHub Projects:

Let's Explore the Project ribbon in GitHub:

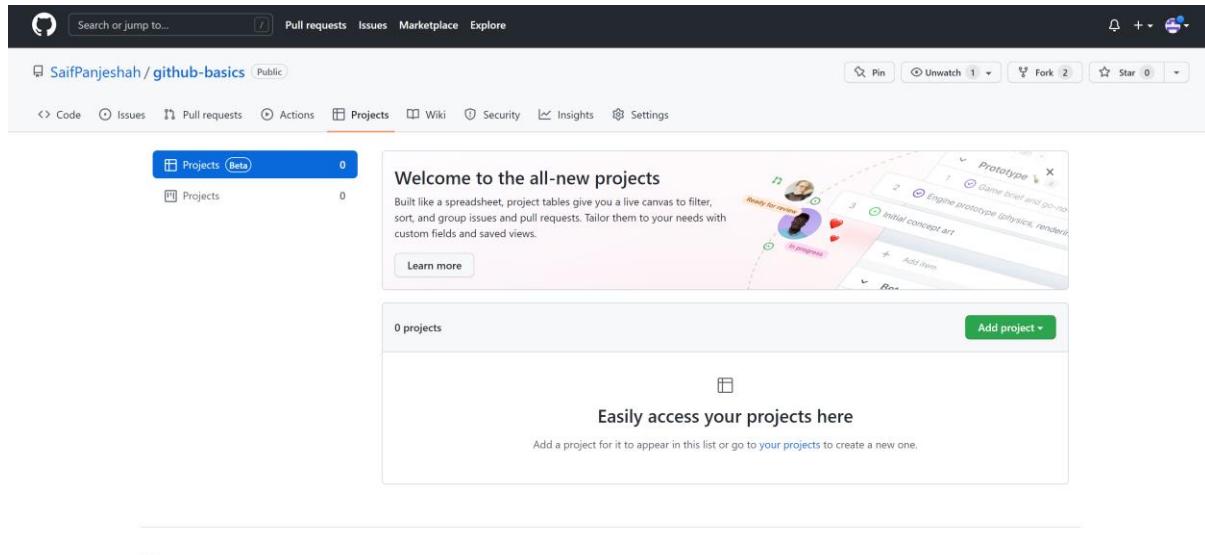


Fig. Project in GitHub

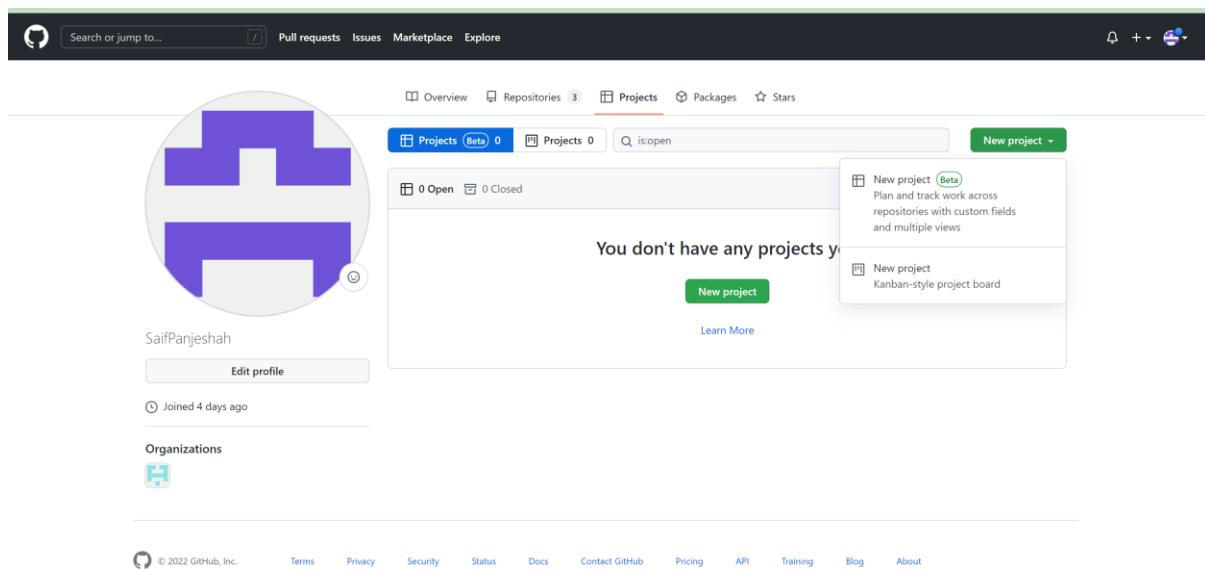


Fig. Creating a Project with Kanban Style

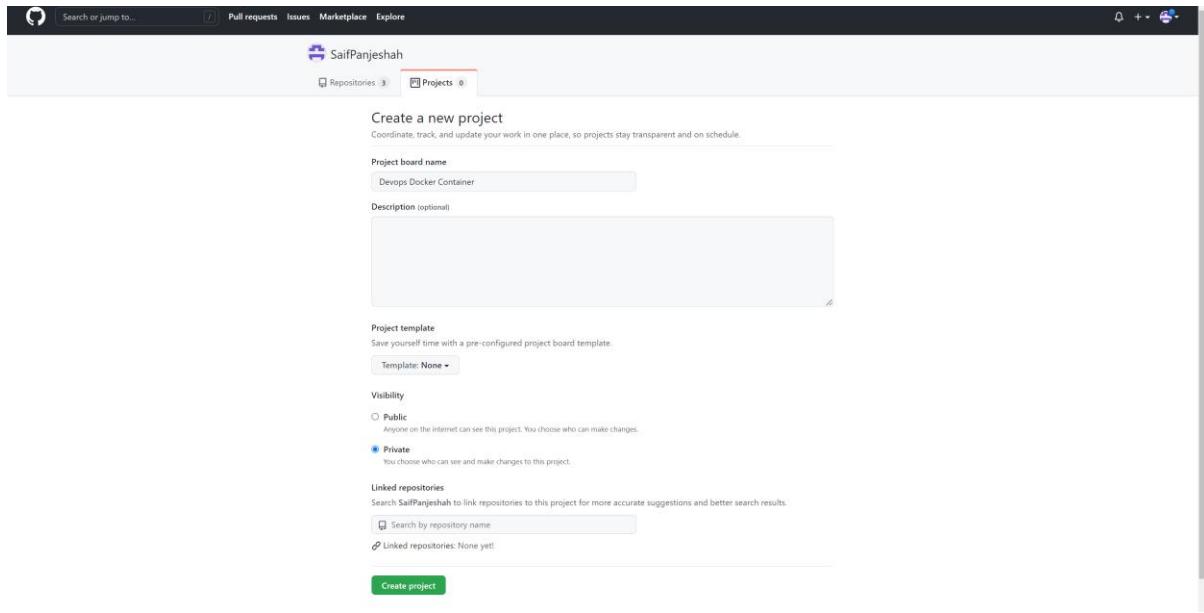


Fig. Devops Docker Container Project

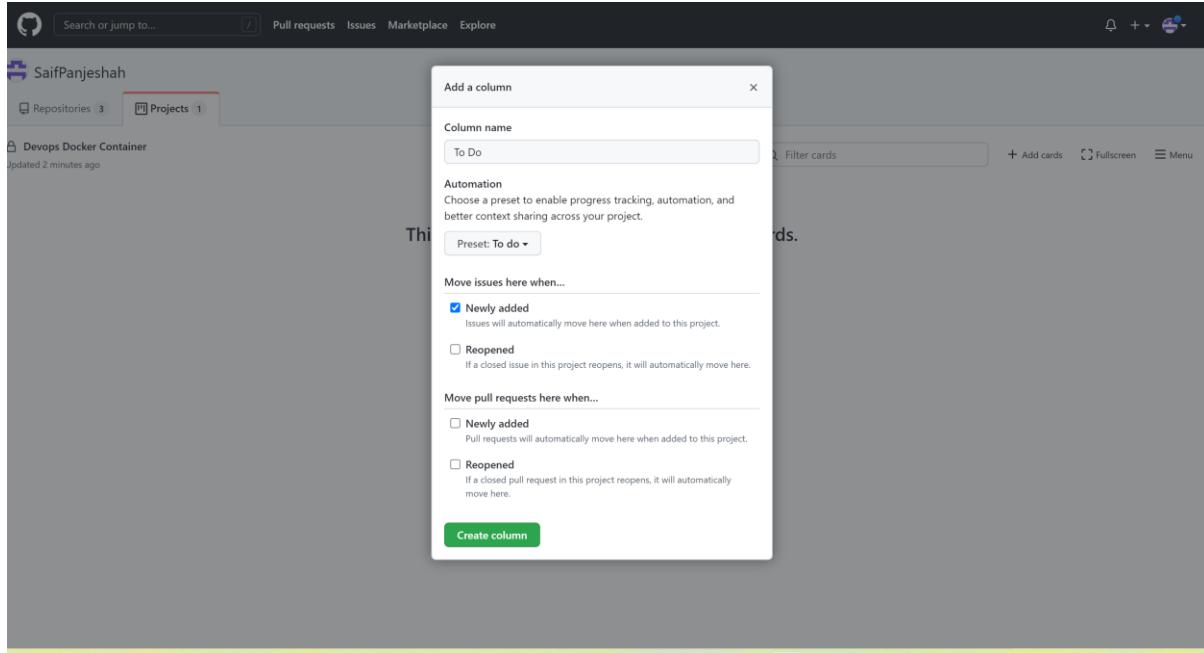


Fig. adding a column what to do

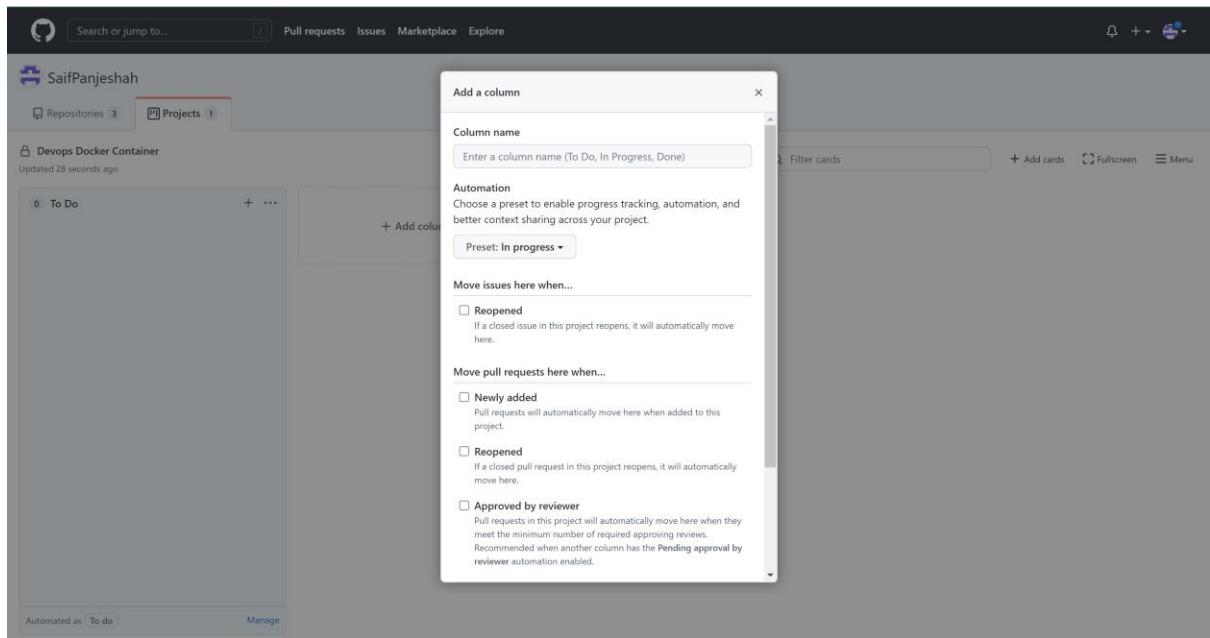


Fig. Creating another column with InProgress

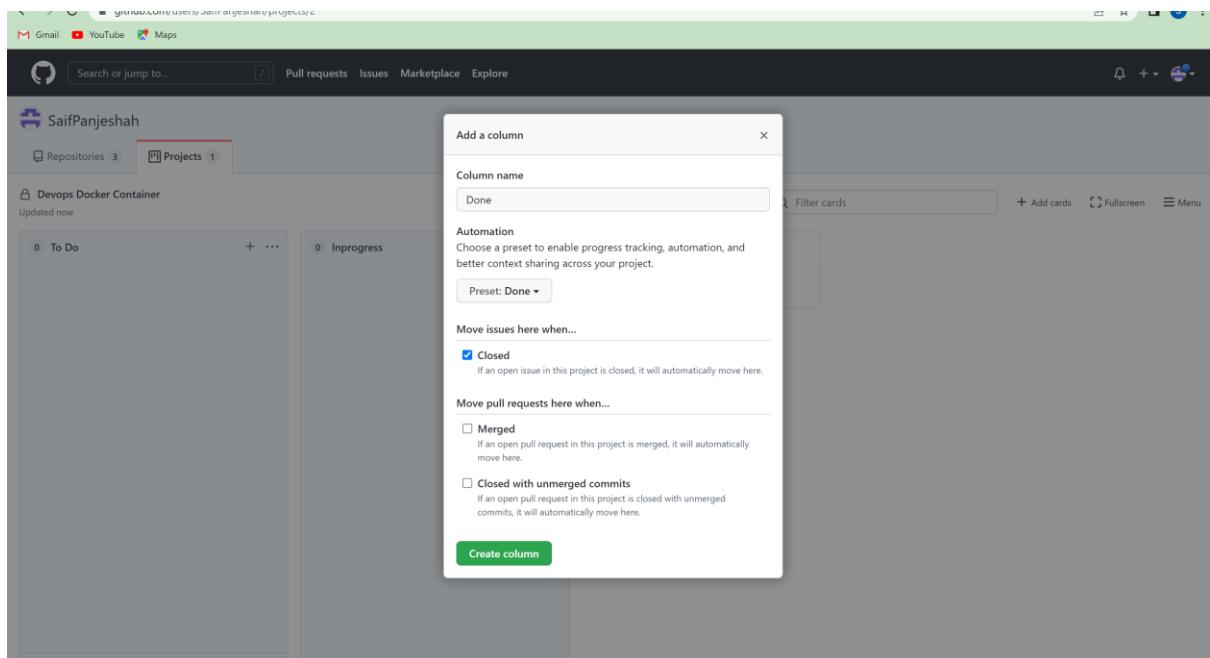


Fig. Final Column with work Done

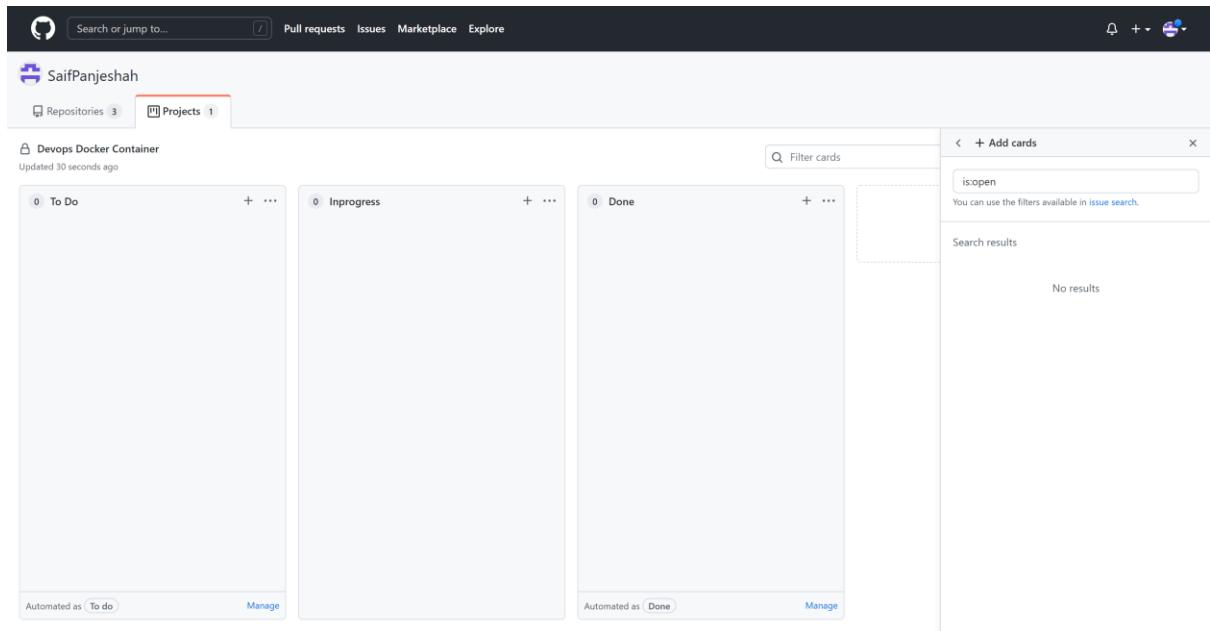


Fig. Project Created successful

Let's assign a work with open Issues on this Project:

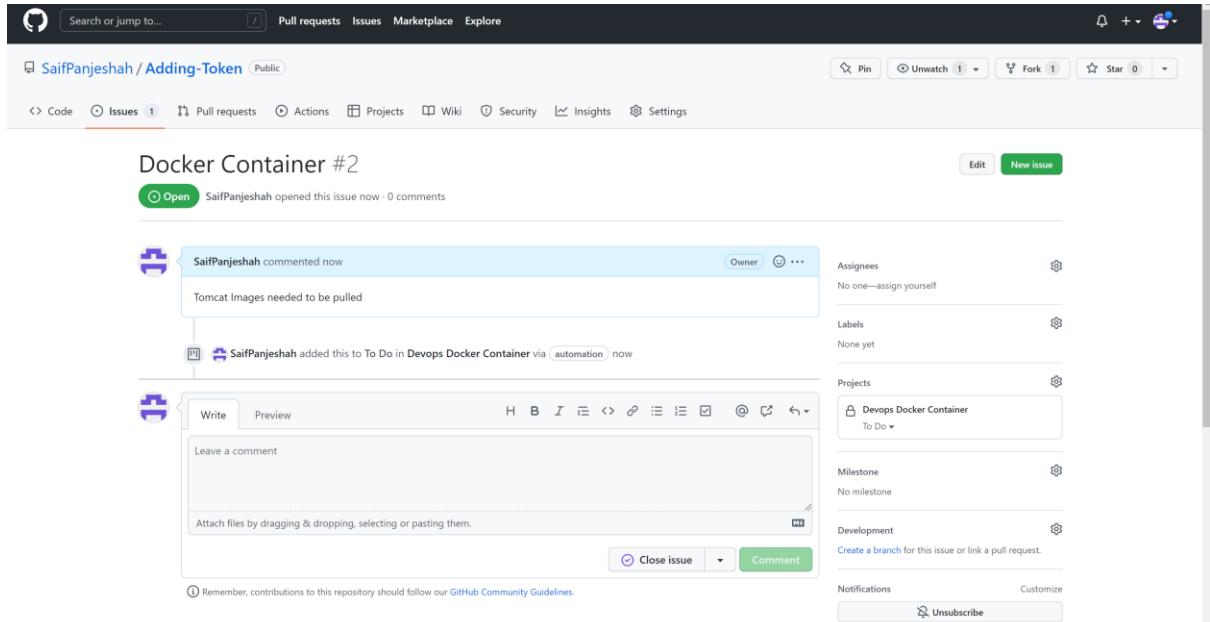


Fig. Creating new Issues

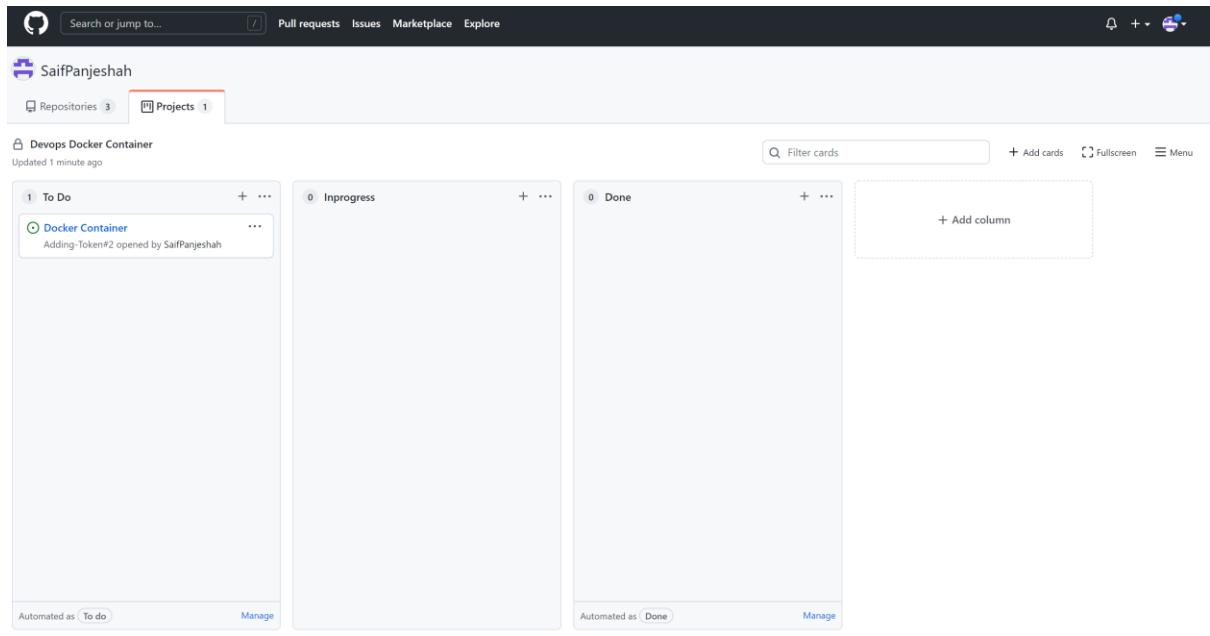


Fig. Card has been successful added to projects

Close this Issue and create a new Issues Check again the Devops Docker Container Project

A screenshot of an issue detail page for 'Docker Container #2'. The top bar shows 'Edit' and 'New issue'. The issue is marked as 'Closed' by 'SaifPanjeshah' 5 minutes ago with 1 comment. The timeline shows: 'Tomcat Images needed to be pulled' (comment by SaifPanjeshah 5 minutes ago); 'SaifPanjeshah added this to To Do in Devops Docker Container via automation 4 minutes ago'; 'SaifPanjeshah closed this as completed now'; 'SaifPanjeshah reopened this now'; 'SaifPanjeshah reopened this 28 seconds ago'; 'SaifPanjeshah commented now' (comment by SaifPanjeshah now); 'Images pull in Initial Commit' (comment by SaifPanjeshah now); and 'SaifPanjeshah closed this as completed now'. The right sidebar includes sections for 'Assignees' (No one—assign yourself), 'Labels' (None yet), 'Projects' (Devops Docker Container), 'Milestone' (No milestone), 'Development' (Create a branch for this issue or link a pull request), 'Notifications' (Customize, Unsubscribe), and '1 participant' (SaifPanjeshah). There are also buttons for 'Lock conversation' and 'Pin issue'.

Fig. Closing Issue

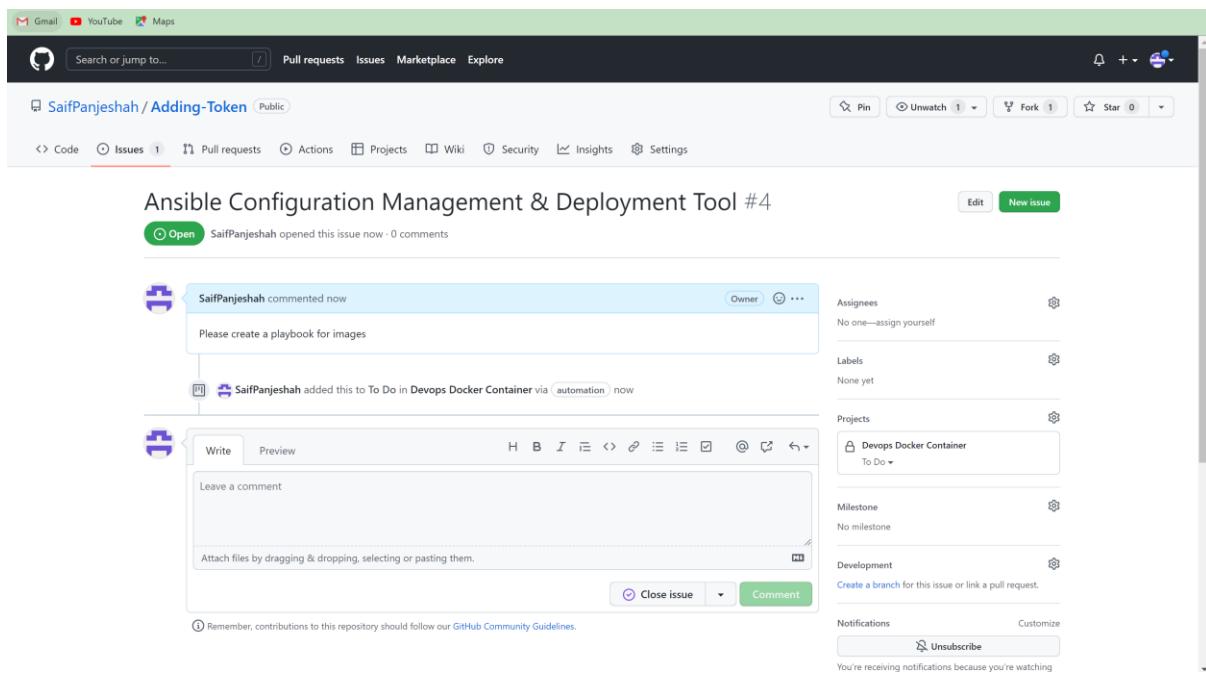


Fig. Creating New Issue

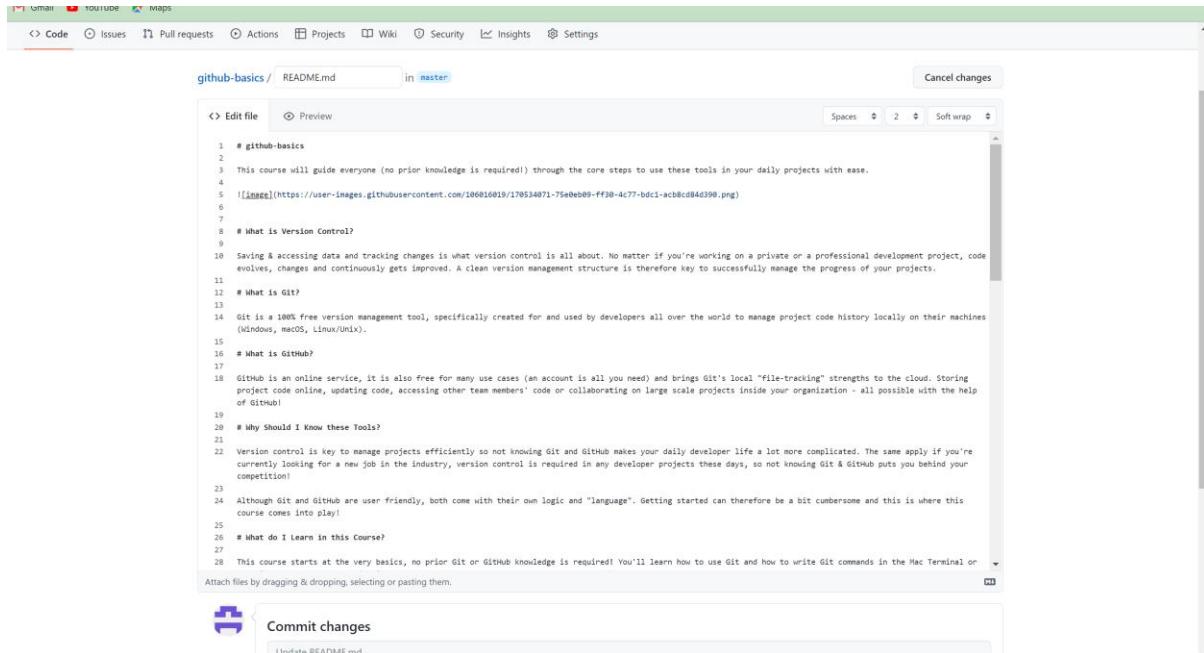
Let's Explore Devops Docker Container Project:

The screenshot shows a GitHub project board for 'Devops Docker Container'. It features three columns: 'To Do', 'In progress', and 'Done'. The 'To Do' column has one card: 'Ansible Configuration Management & Deployment Tool'. The 'In progress' column has one card: 'Adding-Token#2'. The 'Done' column has one card: 'Docker Container'. A new card 'Adding-Token#2' is being added to the 'In progress' column. The board includes a search bar, filter options, and a 'Manage' button.

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 creating a README file. At the top, there are navigation links: Gmail, YouTube, Maps, Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below this, the repository path is shown as `github-basics / README.md` in the master branch. The main area contains the following text:

```
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. Storing 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
```

At the bottom, there is a "Commit changes" button and a "Update README.md" link.

Fig. Creating a README File

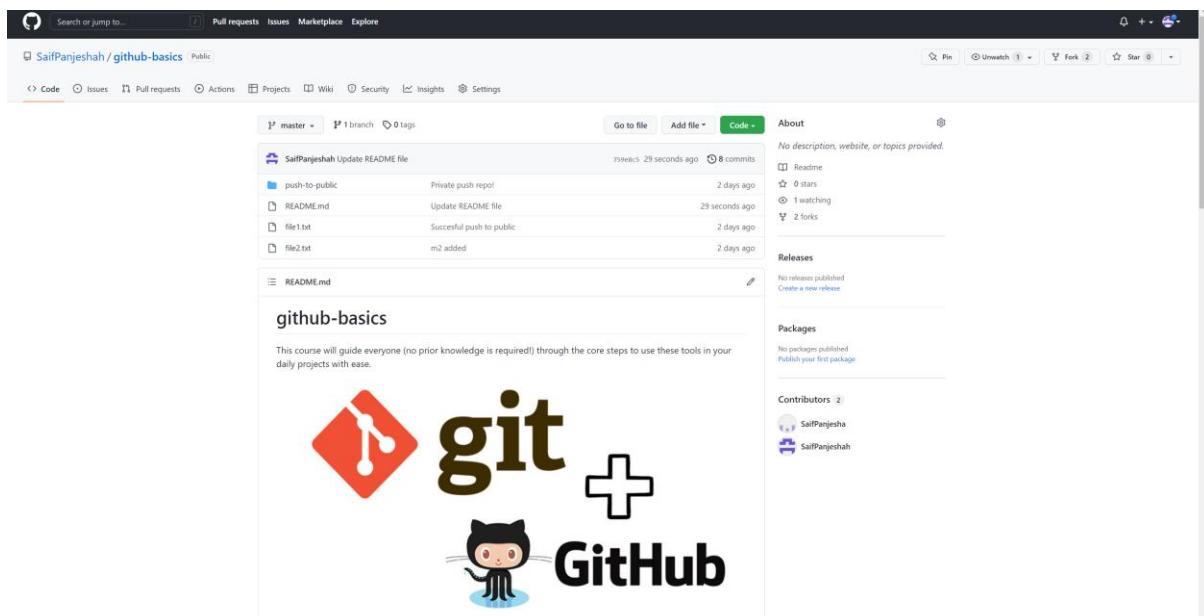


Fig. Complete README File added

Presenting Yourself as a Developer on GitHub:

Making GitHub Page more attractive.

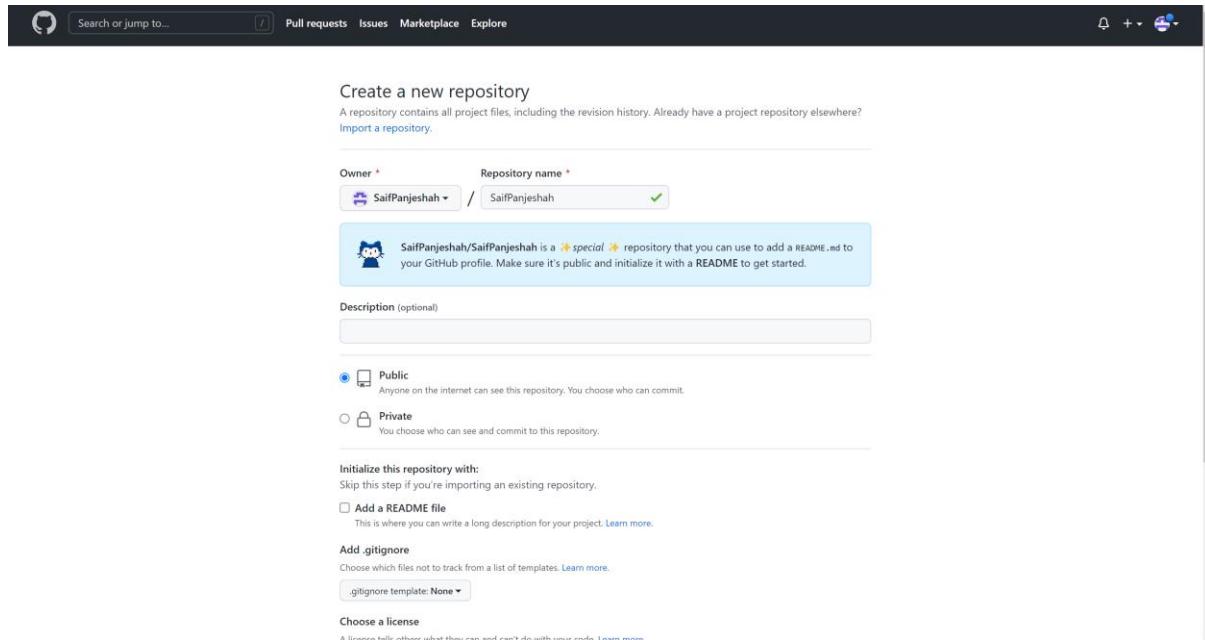


Fig. Creating Special secret repository with owner name

The screenshot shows the GitHub user profile for "Saif Panjeshah". The profile includes a circular profile picture, the name "Saif Panjeshah", and the handle "SaifPanjeshah". Below the profile is an "Edit profile" button. To the left, there are links for "Pune", a LinkedIn profile link, and "Joined 4 days ago". Under "Organizations", there is a small icon. The main area shows the user's repositories: "github-basics" (Public, 2 stars), "gitHub1-basics" (Public, 2 stars), "Adding-Token" (Public, 1 star), and "SaifPanjeshah" (Public, 0 stars). A sidebar on the right allows users to "Customize your pins". The entire screenshot is framed by a light gray border.

Fig. Presenting Yourself as a Developer on GitHub

About GitHub Stars:

Stars are all about likes on social media “Git and GitHub Basics”.

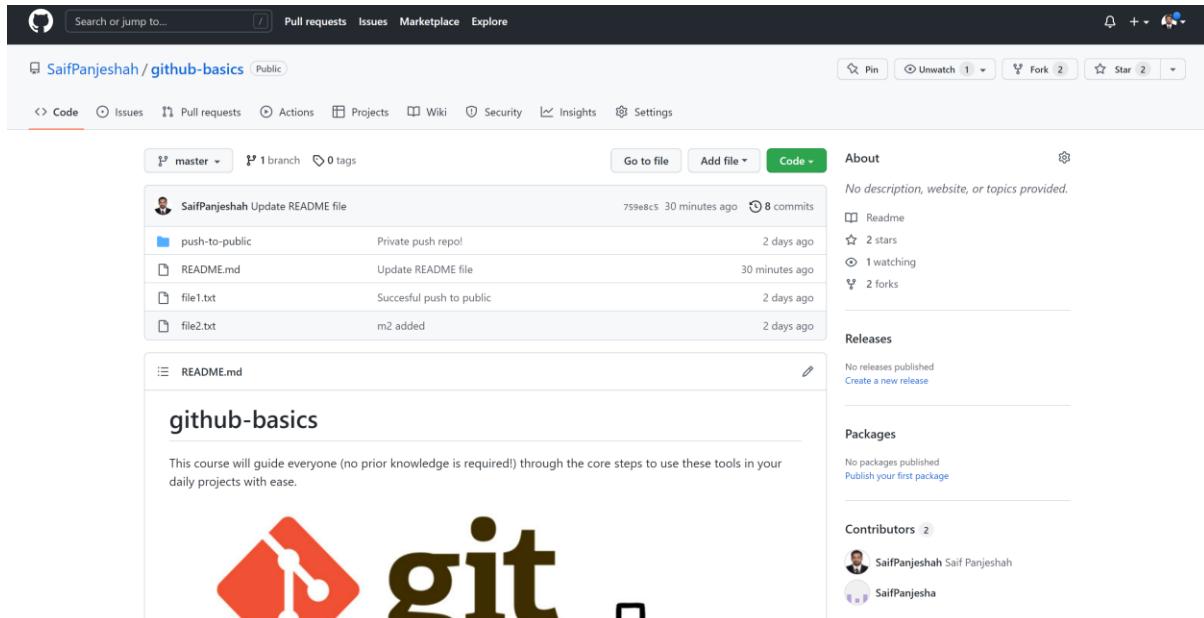


Fig. GitHub Stars

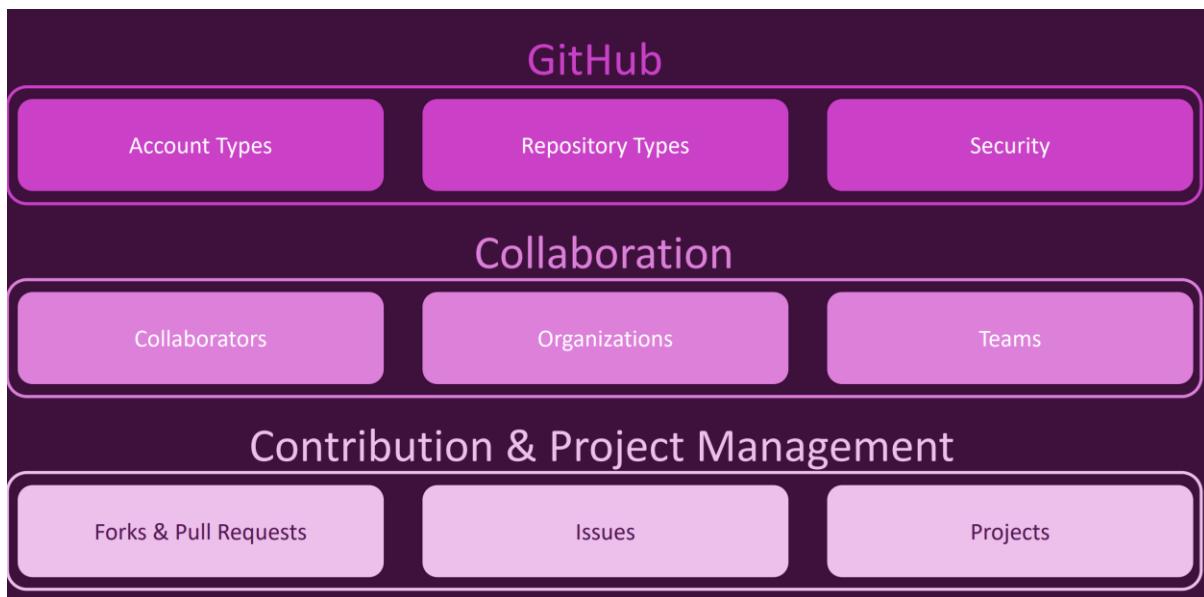


Fig. Wrap up Model Summary

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>