

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

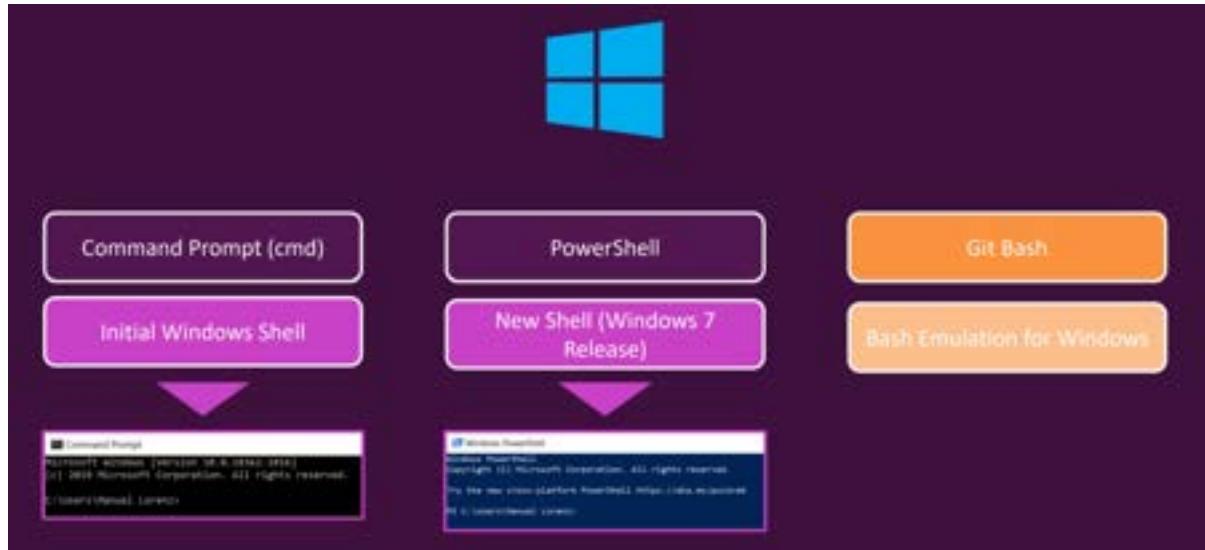


Fig. Windows Terminology

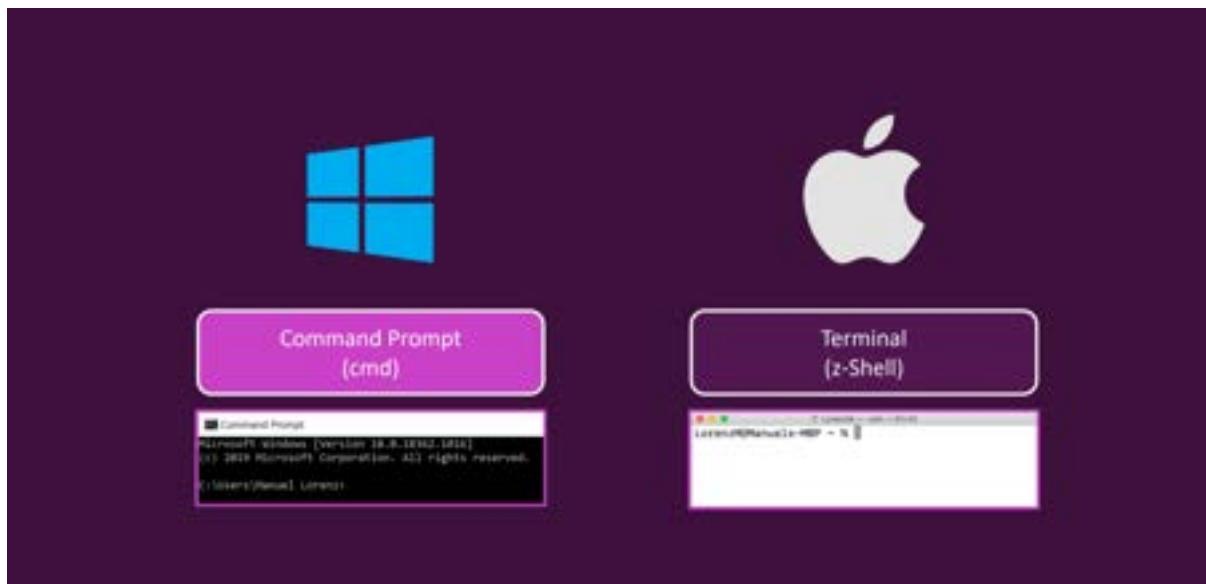


Fig. Command Line Tools

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

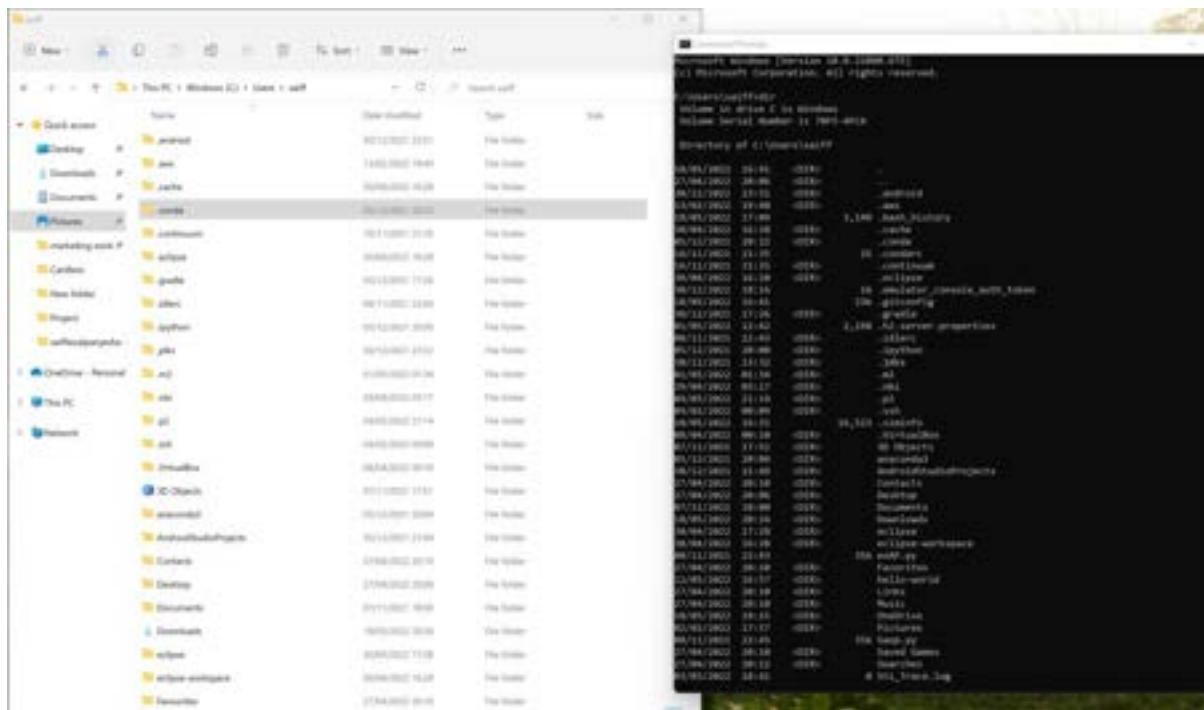


Fig. Lists of Items

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

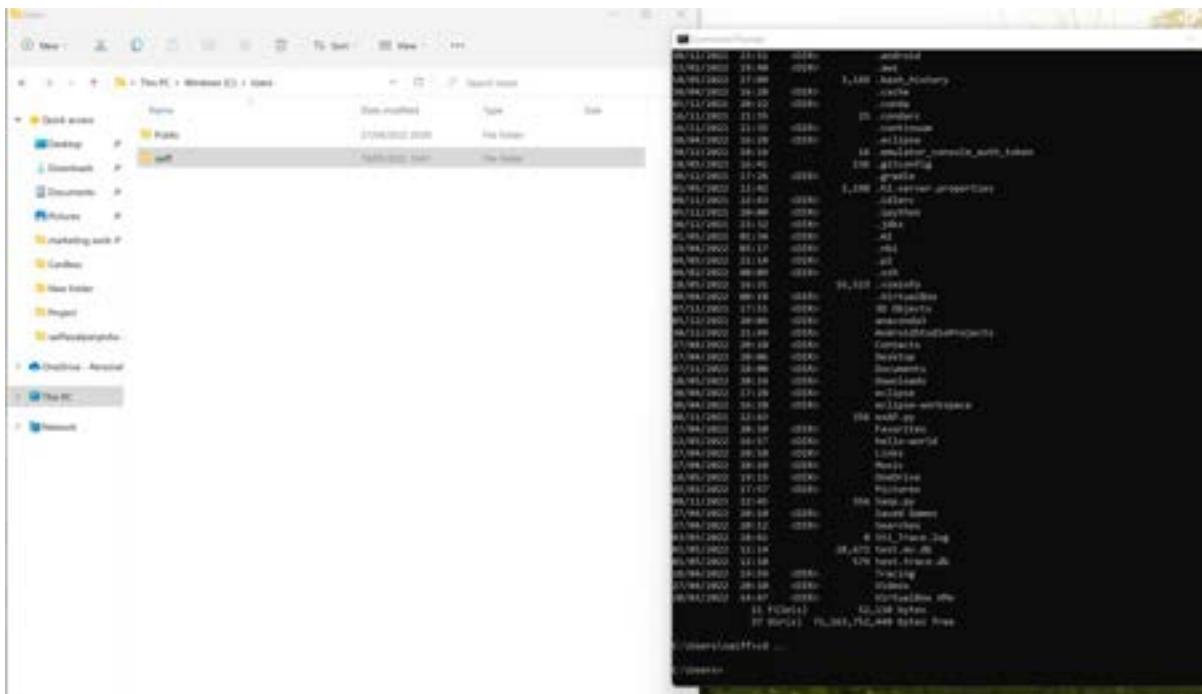


Fig. Change Directory to Users

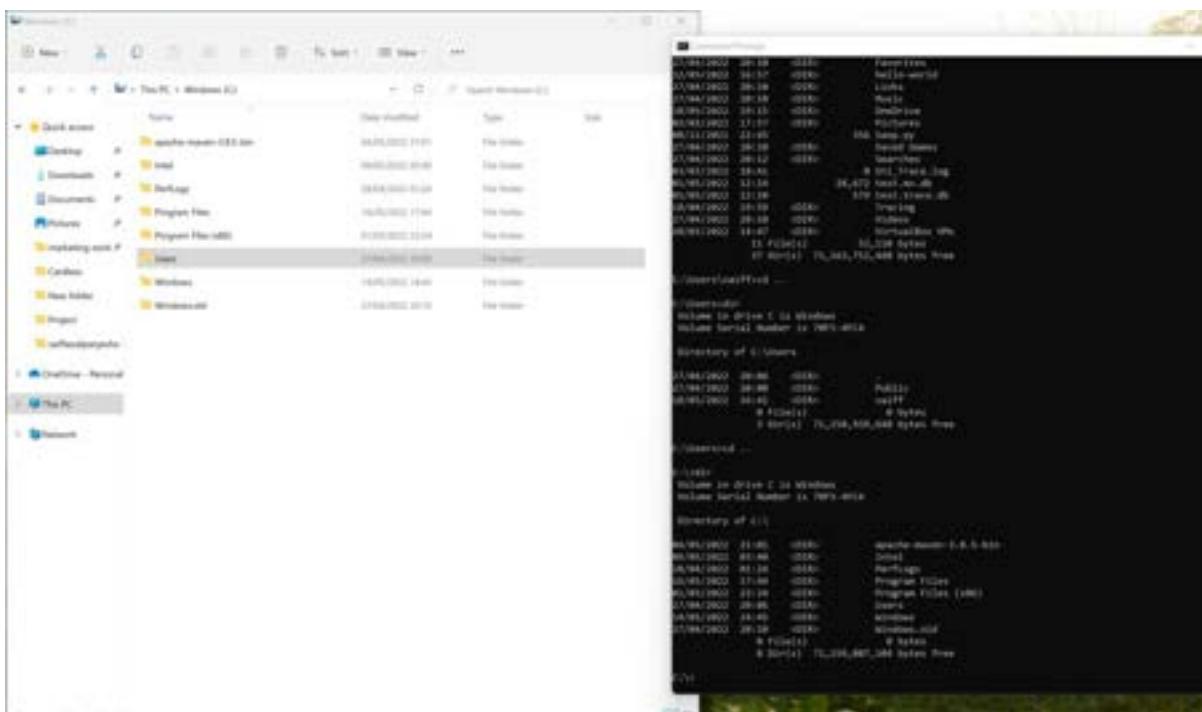


Fig. Change Directory to C drive

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
```

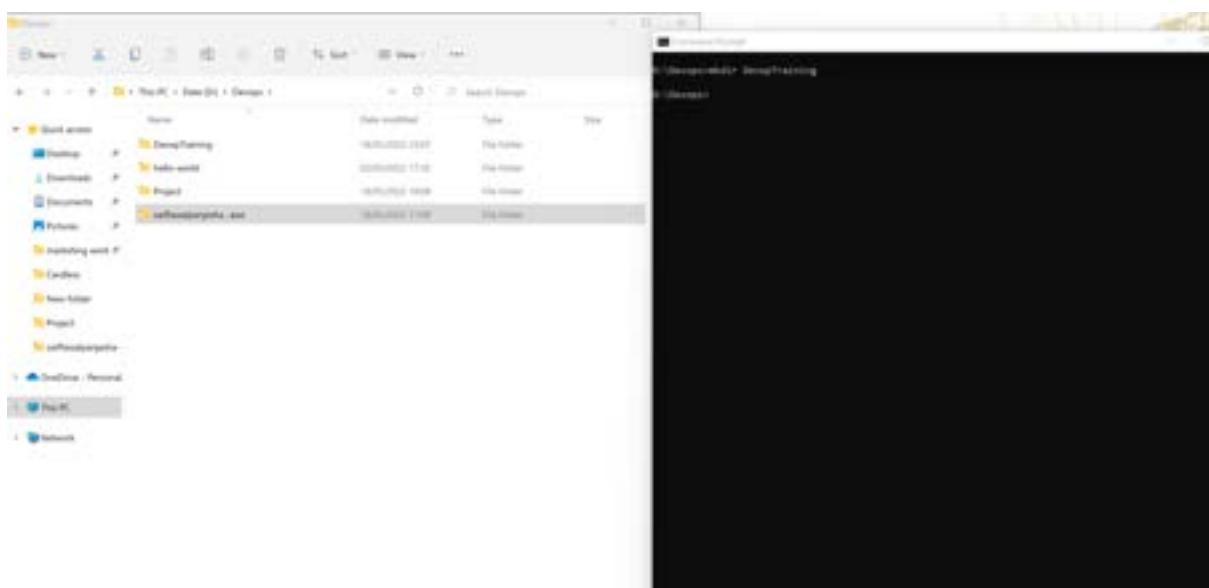
```
D:\Devops\saiffaizalpanjesha -aws>dir  
Volume in drive D is Data  
Volume Serial Number is 04F5-839B  
Directory of D:\Devops\saiffaizalpanjesha -aws  
18/05/2022 17:49 <DIR> .  
18/05/2022 18:08 <DIR> ..  
12/04/2022 08:49 26,740 3646_Skills.pdf  
18/05/2022 17:49 252 Devops Other Courses.txt  
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>          saiffaizalpanjeshaw-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 #4F5-8396

Directory of D:\Devops

18/05/2022  21:23    <DIR>          .
02/05/2022  17:35    <DIR>          hello-world
18/05/2022  23:23    <DIR>          main
18/05/2022  21:23    <DIR>          others
18/05/2022  18:08    <DIR>          Project
18/05/2022  17:49    <DIR>          saiffaizalpanjehsa -www
18/05/2022  21:23           13 test2.txt
                           1 File(s)      13 bytes
                           6 Dir(s)  275,885,586,560 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>move test2.txt
D:\Devops>cd main
Volume in drive D is Data
Volume Serial Number is #4F5-8396

Directory of D:\Devops\main

18/05/2022  21:30    <DIR>          .
18/05/2022  21:24    <DIR>          # File(s)      0 bytes
                           2 Dir(s)  275,885,375,362 bytes free

D:\Devops\main>.

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

D:\Devops>cd ..
Volume in drive D is Data
Volume Serial Number is #4F5-8396

Directory of D:\Devops

18/05/2022  21:31    <DIR>          .
02/05/2022  17:35    <DIR>          hello-world
18/05/2022  21:30    <DIR>          main
18/05/2022  21:21    <DIR>          others
18/05/2022  18:08    <DIR>          Project
18/05/2022  17:49    <DIR>          saiffaizalpanjehsa -www
                           0 File(s)      0 bytes
                           6 Dir(s)  275,885,375,362 bytes free

D:\Devops>cd main
Volume in drive D is Data
Volume Serial Number is #4F5-8396

Directory of D:\Devops\main

18/05/2022  21:31    <DIR>          .
18/05/2022  21:31    <DIR>          .
18/05/2022  21:21           13 test2.txt
                           1 File(s)      13 bytes
                           2 Dir(s)  275,885,375,362 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?

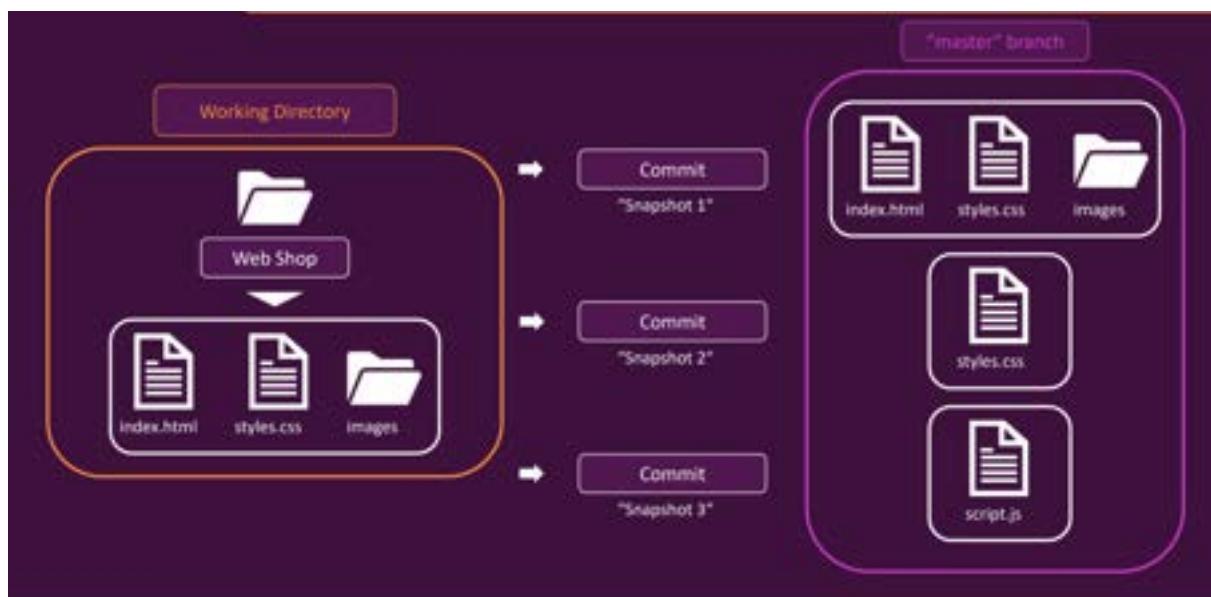


Fig. Working of Git.

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.”

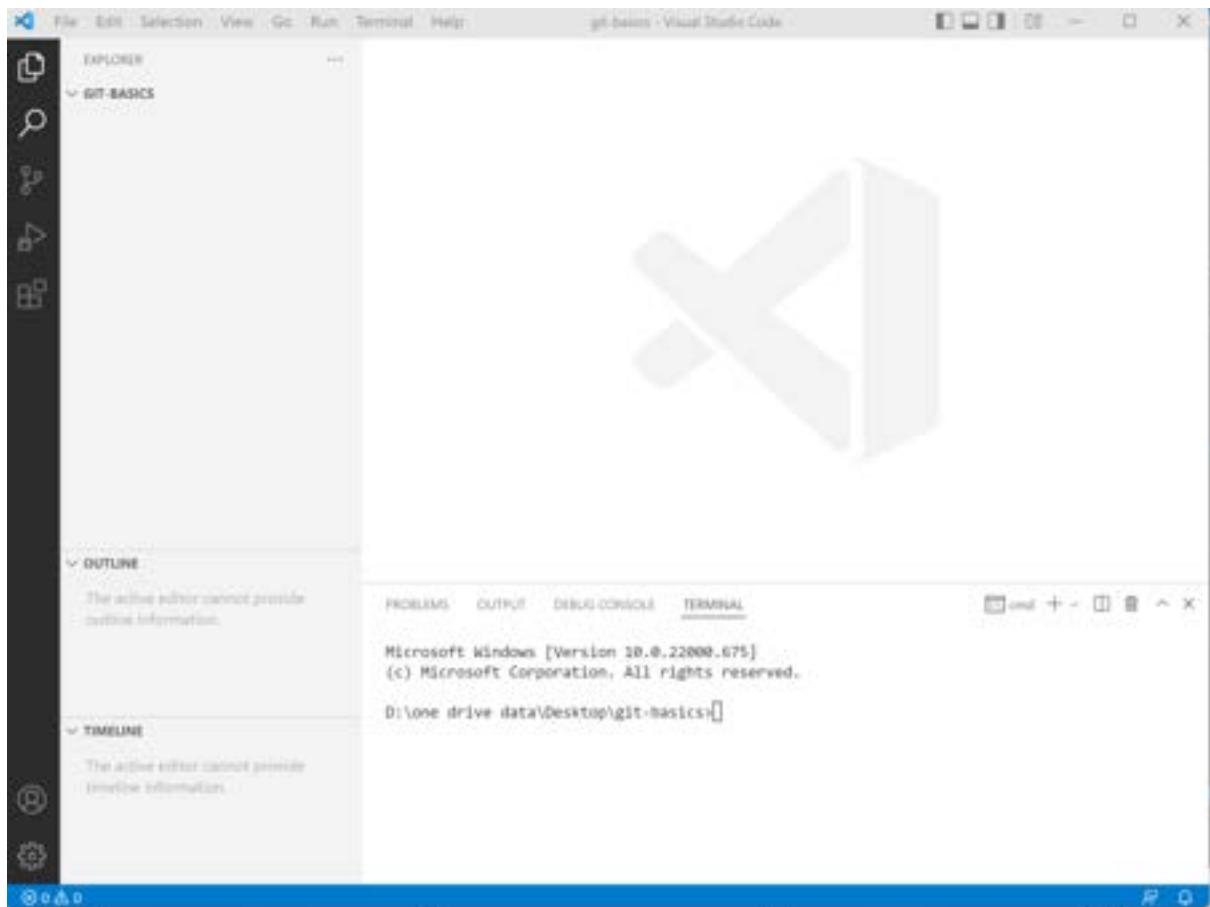


Fig. Visual Studio Install Success

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/
```

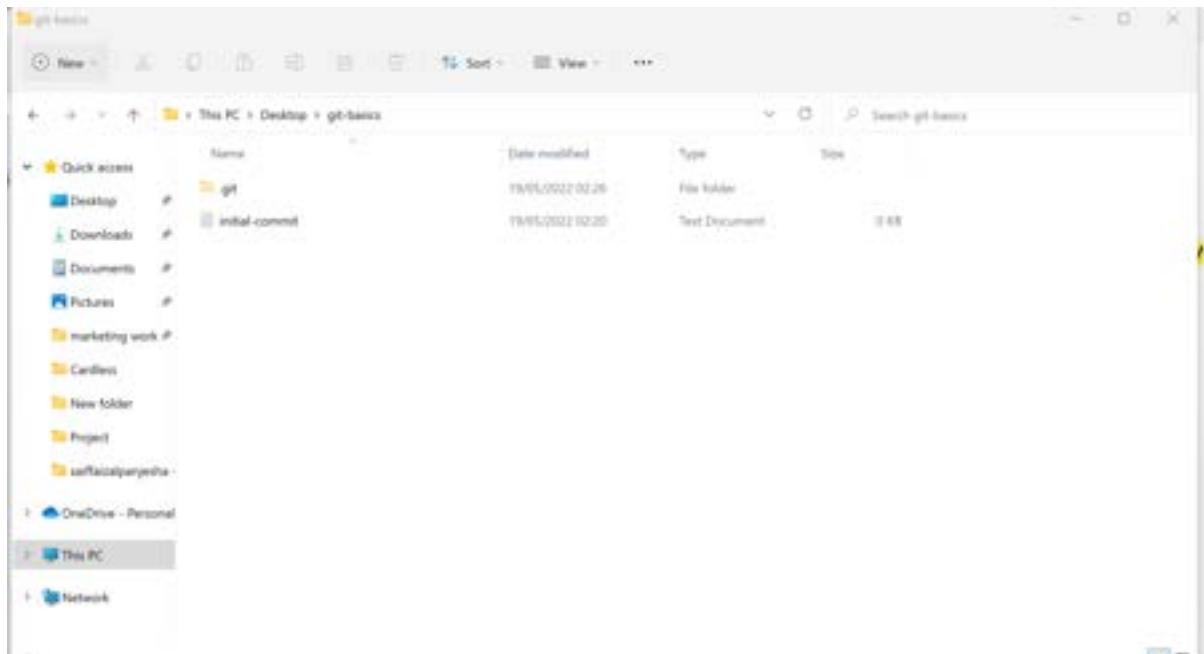


Fig. git hidden folder created after initialized

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>
```

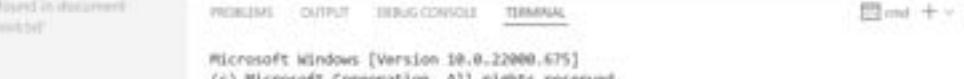
```
Date: Thu May 19 02:36:24 2022 +0530
```

```
updated file.txt
```

Create a second Commit File:



Fig. Second Commit in git



A screenshot of a terminal window titled "OUTLINE". The terminal shows the following output:

```
Microsoft Windows [Version 10.0.22996.675]
(c) Microsoft Corporation. All rights reserved.

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

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

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

Fig. git status for untracked file

The screenshot shows a Visual Studio Code interface with the title bar "File Edit Selection View Go Run Terminal Help * second-commit.txt - git-basics : Visual Studio Code". The left sidebar has sections for "EXPLORER", "GIT-BASICS", and "OUTLINE". The "GIT-BASICS" section lists two files: "initial-commit.txt" and "second-commit.txt". The main area is a terminal window titled "Terminal" with the command "git log" running. The output shows:

```
D:\OneDrive\Documents\Desktop\git-basics>git log
```

commit 1e3d15a (HEAD → master)
Author: [REDACTED] <[REDACTED]>
Date: Mon Oct 10 14:44:16 2022 +0200

 initial commit

D:\OneDrive\Documents\Desktop\git-basics>

Fig. "git add" to track file

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

```
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 dc4a67141cb3c3529cdc7da6200e96e0f0c56387 (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
```

Fig. "git log "jump back to history"

To check Back the previous commit

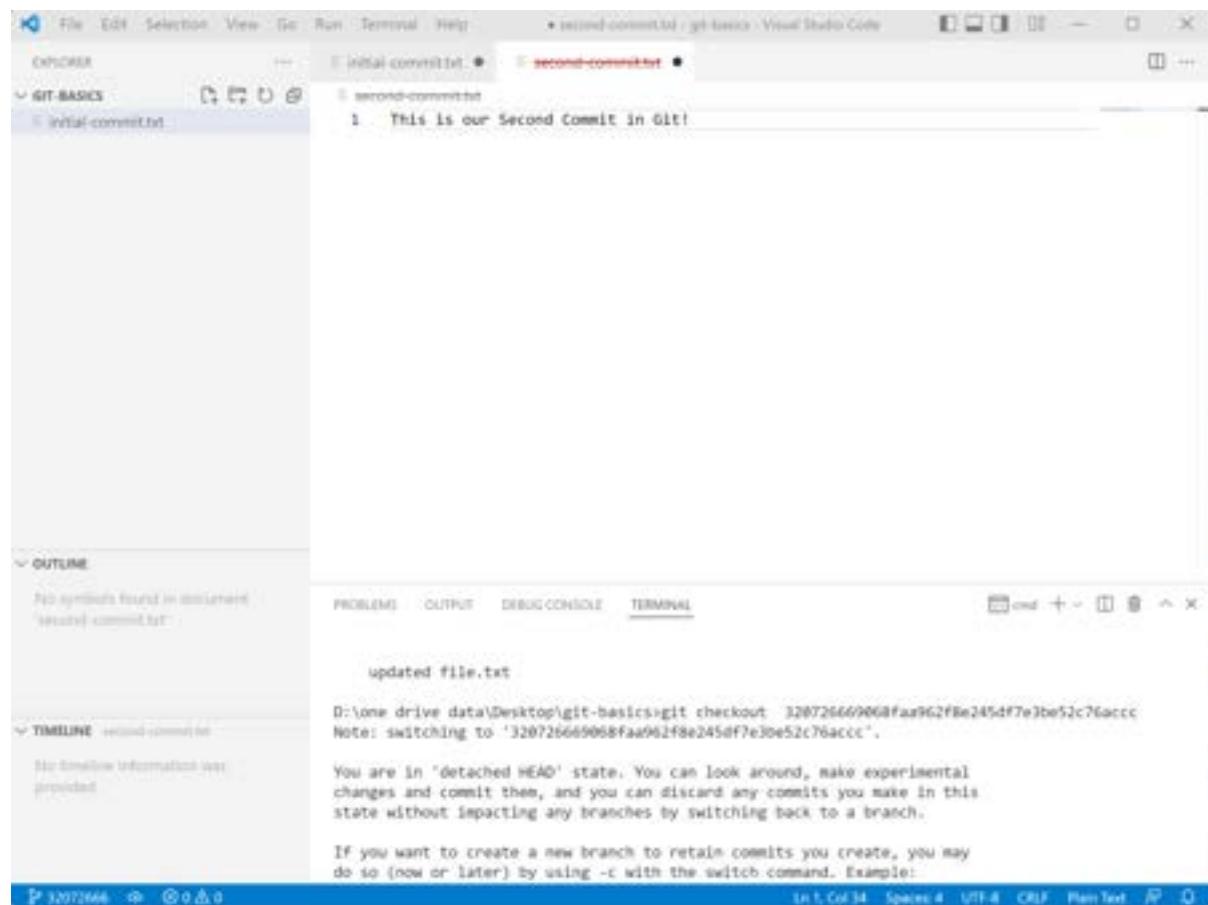


Fig. "Checkout our initial commit"

```
HEAD is now at 3207266 updated file.txt
D:\One Drive\data\Desktop\git-basics>git log
commit 32072669008faa962f9e245df7e3be52c7baacc (HEAD)
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date:   Thu May 19 02:36:24 2022 +0530

    updated file.txt
```

Fig. Not updated the second commit in history

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

The Answer is No not deleted.

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

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

D:\One Drive\data\Desktop\git-basics>git log
commit d44d7843cb3c8539dc7da620e0040f6c56187 (HEAD -> master)
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date:   Thu May 19 15:02:23 2022 +0530

    added second-commit.txt

commit 32072669008faa962f9e245df7e3be52c7baacc
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>
```

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 the Visual Studio Code interface. The 'TERMINAL' tab is active, showing the command line output of 'git branch'. The output shows one branch named 'MASTER' preceded by an asterisk (*). The terminal window title is 'second-commit.txt - git basics - Visual Studio Code'. The 'EXPLORER' sidebar on the left shows a folder named 'GIT-BASICS' containing 'initial-commit' and 'second-commit'. The 'OUTLINE' sidebar shows a file named 'second-commit.txt'. The 'TIMELINE' sidebar shows a commit for 'second-commit.txt' with the message 'added second-commit.txt... 13 lines'.

```
D:\One Drive\data\Desktop\git-basics>git branch
* 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 command-line session:

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

D:\One Drive\data\Desktop\git-basics>git branch Landing-Page

D:\One Drive\data\Desktop\git-basics>git branch
  Landing-Page
* master

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

Fig. Second Branch is Created

For accessing Second Branch git checkout branch-name:

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

```
D:\One Drive\data\Desktop\git-basics>git branch
  Landing-Page
* master

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 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 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 three items: 'Initial-commit.txt', 'second-commit.txt', and 'Working_with_Branches.txt'. The main area is a terminal window with the following text:

```
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
```

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 dc4a67141cb3c3529cdc7da6280e9600f8c56387 (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

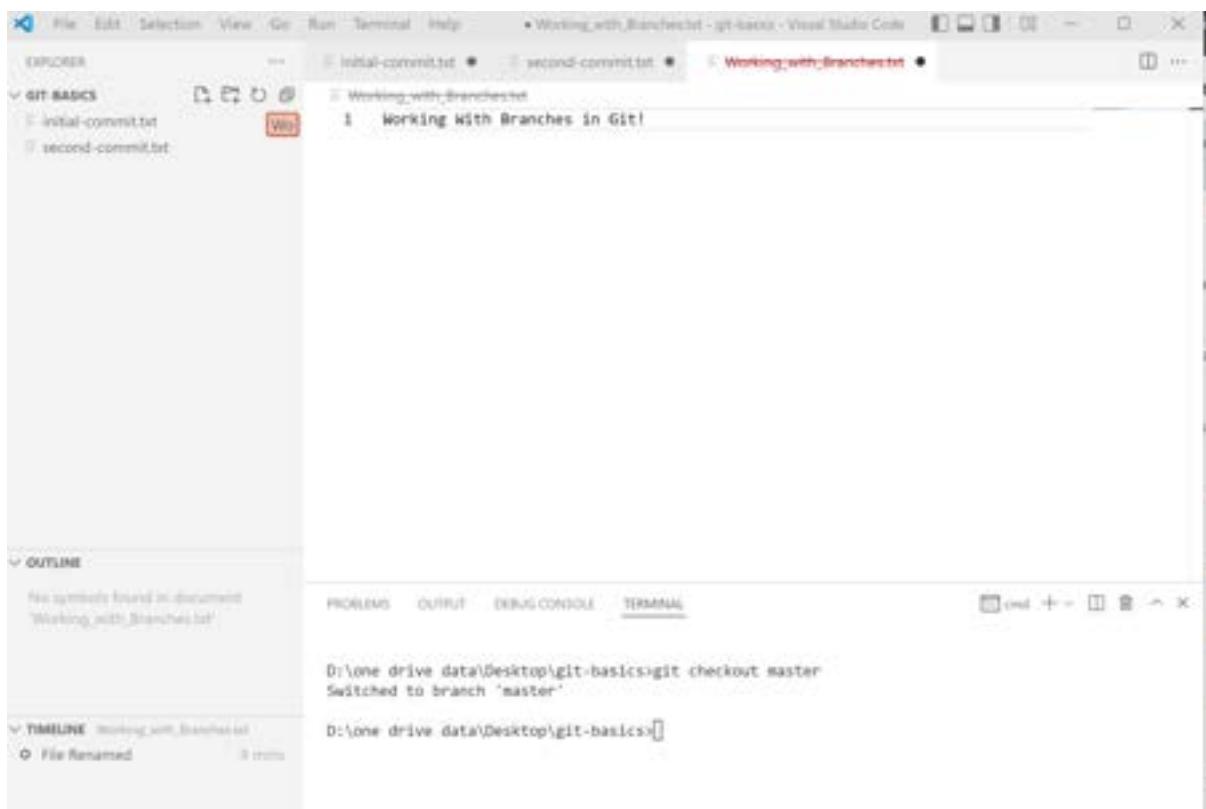


Fig. Switch to Master Branch

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:

- Git History:** Shows three commits: "initial-commit.txt", "second-commit.txt", and "Working_with_Branches.txt".
- Terminal:** Displays the command "git merge quickfix" being run, showing the output:

```
D:\one drive data\Desktop\git-basics>git merge quickfix
Updating dc4a671..c08816c
Fast-forward
  Working_with_Branches.txt | 0
  1 file changed, 0 insertions(+), 0 deletions(-)
  create mode 100644 Working_with_Branches.txt
```

Fig. Merging with Branch

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

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Terminal Tab:** Working with Branches.txt - git-basics - Visual Studio Code
- Explorer:** Shows a tree view with branches: initial-commit, second-commit, and Working_with_Branches.txt (selected).
- Terminal Content:**

```
D:\One Drive\data\Desktop\git-basics>git log
commit c08816c0be74cecbaf4f5995e25b2f90a0b3cb9a3 [HEAD -> master, quickfix]
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date:   Thu May 19 15:44:16 2022 +0530

    Branches are Working

commit dc4a67141ch3c7529cdc7da6200e9000f9c56387 [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 328726669068faa962f8e245df7e3be52c76acc
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>
```
- Outline:** Shows 'No symbols found in document' and 'Working_with_Branches.txt'.
- Timeline:** Working with Branches.txt
 - Branches are Working (1s, 10 min)
 - File Renamed (13 min)

Fig. Success History of Merging.

Understanding the HEAD

The Latest commit is known as HEAD.

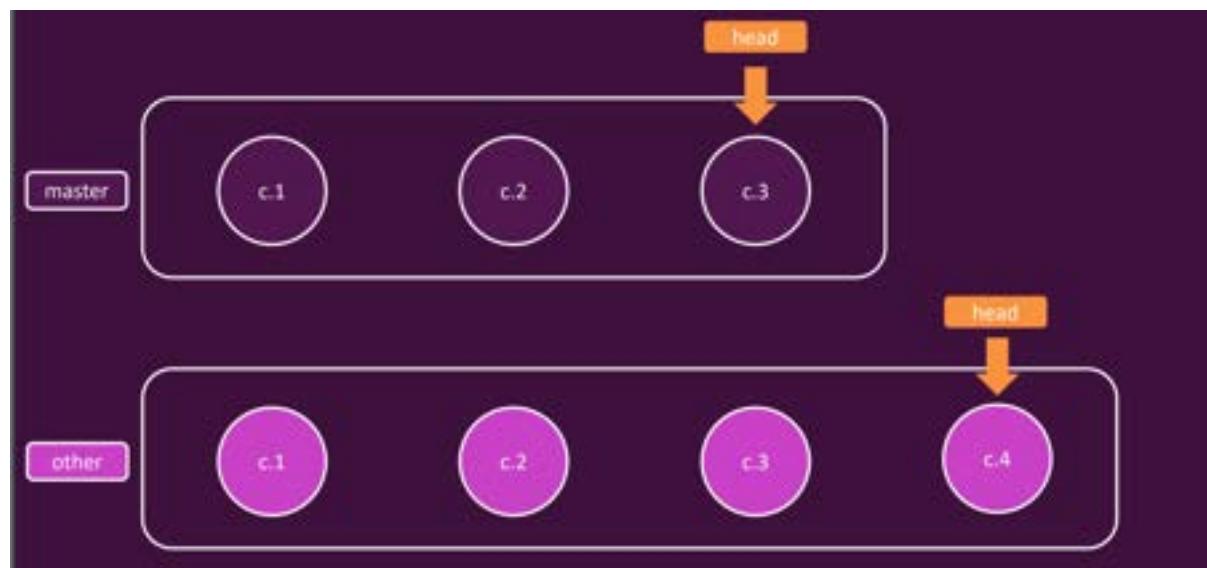


Fig. Head Understanding

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

D:\One Drive Data\Desktop\git-basics>git checkout Landing-Page
Switched to branch 'Landing-Page'

D:\One Drive Data\Desktop\git-basics>git log
commit dca67141cb3c3529cdc7da6200e9600ff0c56387 (HEAD -> 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 320726669868faa962f8e245df7e3be52c76acc
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>[]

Fig. Head Understanding History

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 dca67141cb3c3529cdc7da6200e9600ff0c56387 (HEAD -> 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 320726669868faa962f8e245df7e3be52c76acc
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 master
Switched to branch 'master'

D:\One Drive Data\Desktop\git-basics>git log
commit c08816cobe74cechaaf5995e25b2f9ab3cb93 (HEAD -> master, quickfix)
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date: Thu May 19 15:44:16 2022 +0530

Branches are Working

commit dca67141cb3c3529cdc7da6200e9600ff0c56387 (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 320726669868faa962f8e245df7e3be52c76acc
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>[]

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

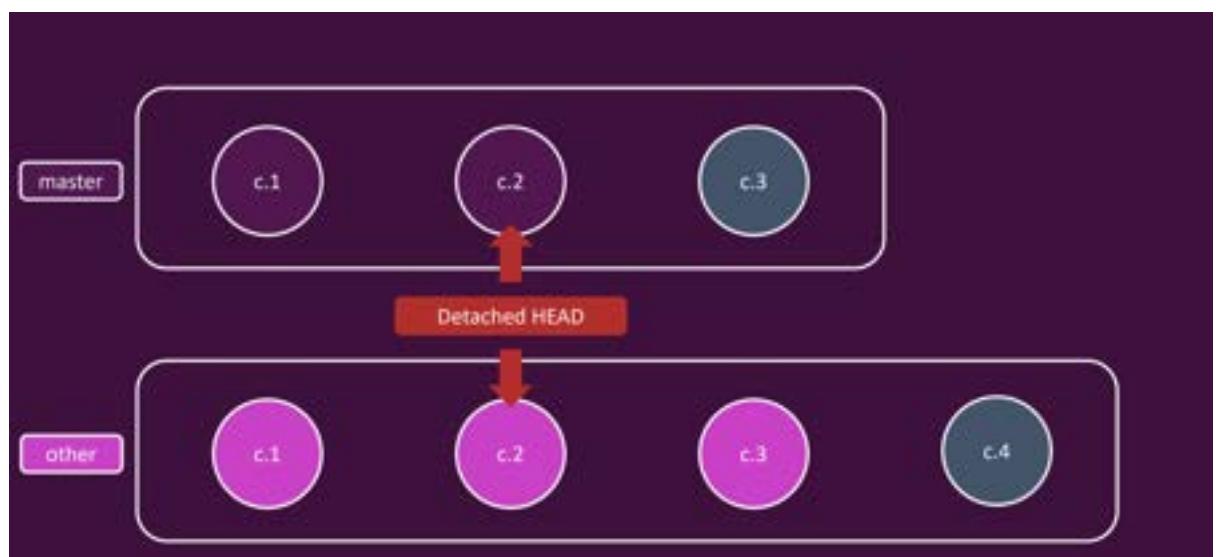


Fig. Detached Head

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

```
commit dc4a67141cb3c3529c4c7da6200e960f0c56387 (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 32872660906ffaa962f8e245df7e1be53c78accc
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 cd8816cd874cecbaf4f5995e25b2f9ab3cb9a3
Note: switching to 'cd8816cd874cecbaf4f5995e25b2f9ab3cb9a3'.

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
```

Fig. Detached Head on terminal

How to avoid this?

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

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

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

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** Working with Branches.txt - git-basics - Visual Studio Code.
- Sidebar:**
 - EXPLORER:** Shows 'GIT-BASICS' with three items: 'initial-commit.txt', 'second-commit.txt', and 'Working_with_Branches.txt'.
 - OUTLINE:** Shows 'No symbols found in document' and 'Working with Branches.txt'.
 - TIMELINE:** Shows 'Working with Branches.txt' with two entries: 'Branches are Working' and 'File Renamed'.
- Terminal:** Shows a command-line session:

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

D:\One Drive\data\Desktop\git-basics>git checkout quickfix
Switched to branch 'quickfix'

D:\One Drive\data\Desktop\git-basics>git branch
  Landing-Page
  master
* quickfix

D:\One Drive\data\Desktop\git-basics>
```
- Bottom Status Bar:** Line 1, Col 30, Spaces: 4, UTF-8, CR/LF, Plain Text, IP, O.

Fig. Avoided Detached Head

Branches & “git switch” (Git 2.2.23)

The “git switch” command

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

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

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

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

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

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

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

The Explorer sidebar on the left shows files: 'initial-commit.txt', 'second-commit.txt', and 'Working_with_Branches.txt'. The Outline sidebar shows a note: '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:

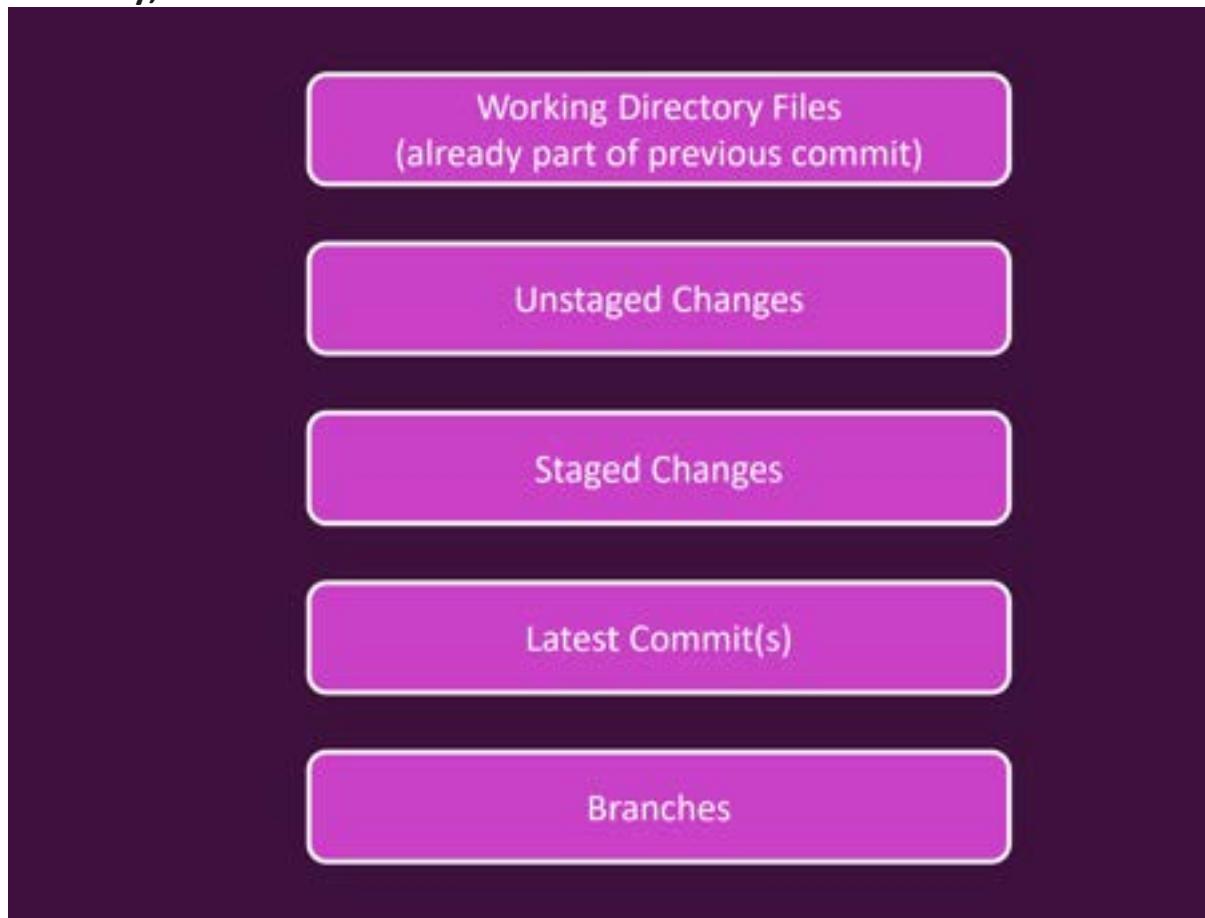


Fig. Deleting Data : Overview

Deleting Working Directory Files:

To check Which files currently on staging area.

git ls-files

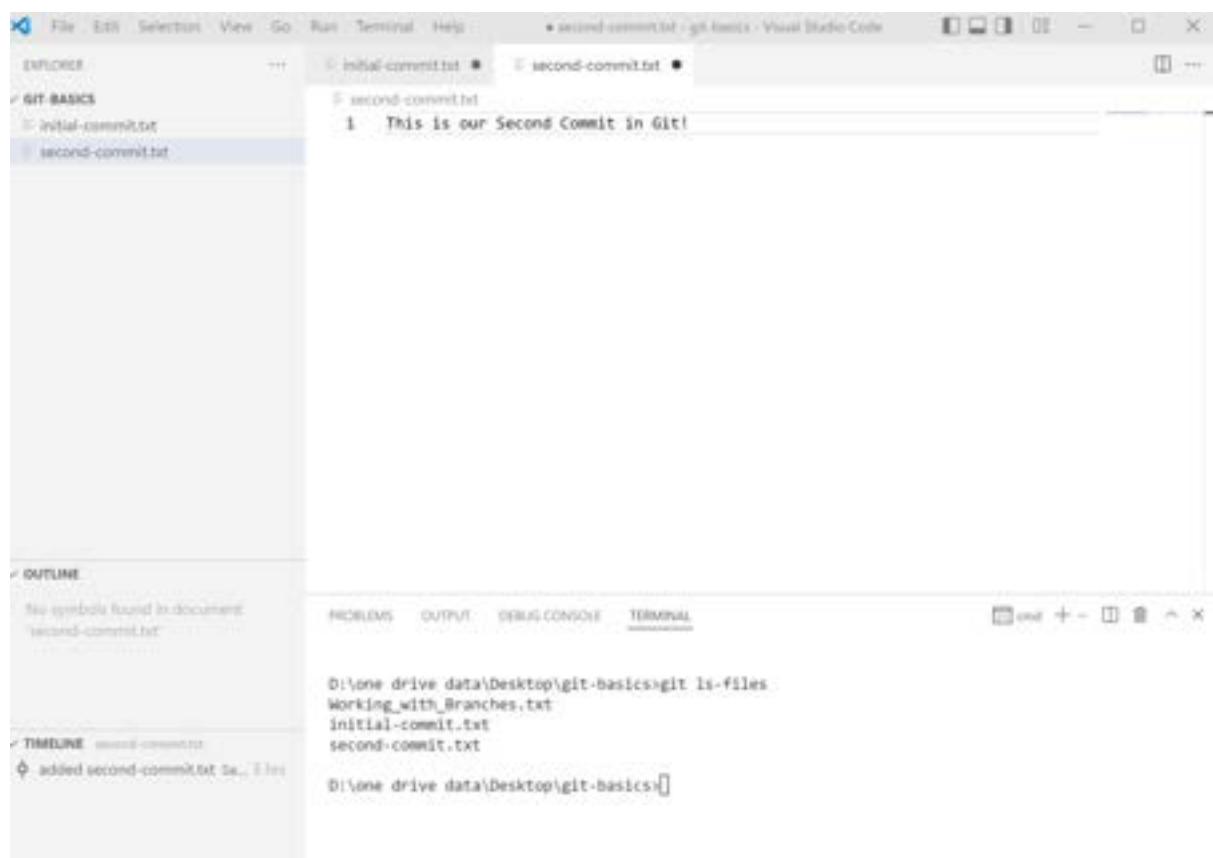


```
D:\One Drive Data\Desktop\git-basics>git status
On branch master
nothing to commit, working tree clean

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



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

- EXPLORER:** Shows two branches: "initial-commit" and "second-commit".
- GIT BASICS:** Shows the same two branches.
- OUTLINE:** No symbols found in document "second-commit.txt".
- TIMELINE:** Shows a commit for "second-commit" and a note: "added second-commit.txt to .gitignore".
- TERMINAL:** Displays the command history:

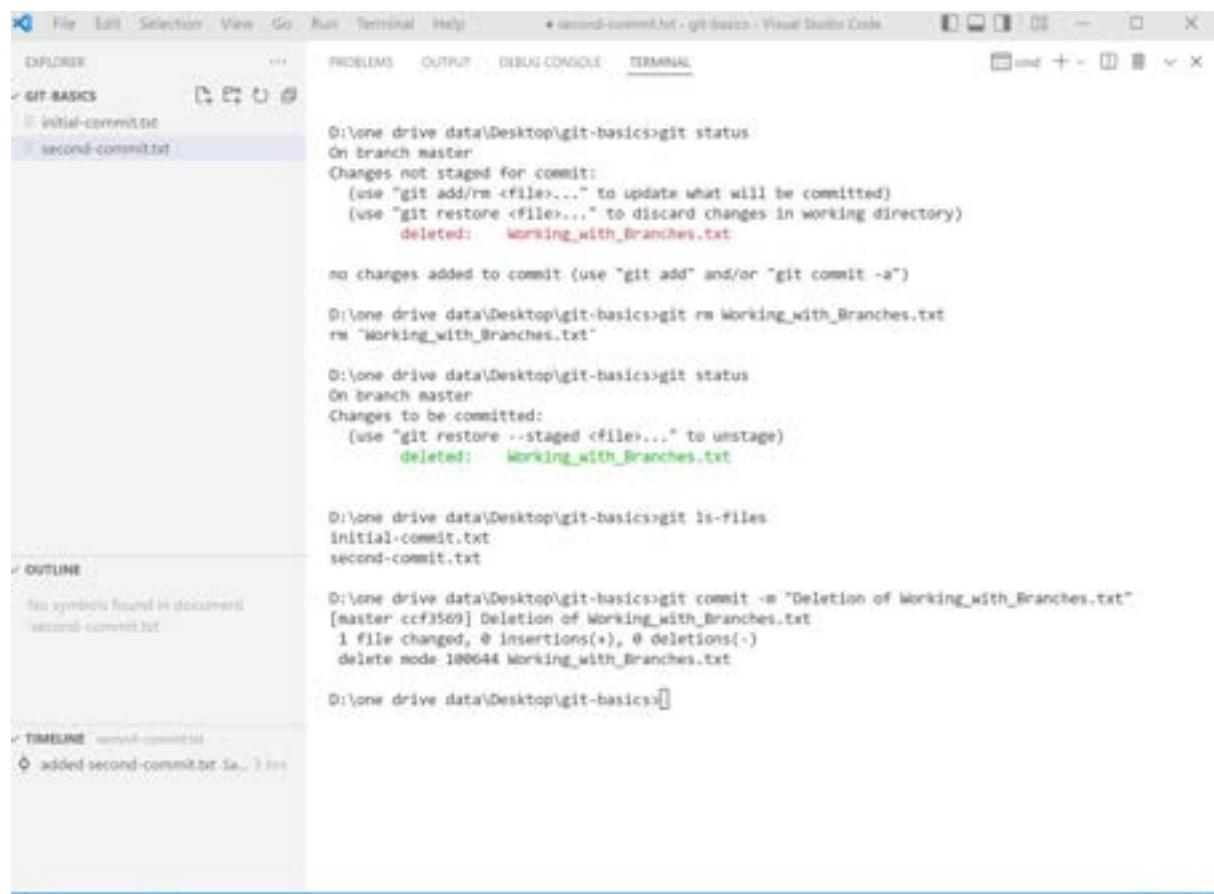
```
D:\One Drive Data\Desktop\git-basics>git status
On branch master
nothing to commit, working tree clean

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. 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 a Visual Studio Code interface with the following details:

- Terminal Tab:** The terminal tab is active, showing a command-line session in the 'git-basics' workspace.
- Output:** The terminal displays the following commands and their results:
 - git status: Shows the file 'Working_with_Branches.txt' is deleted.
 - git rm "Working_with_Branches.txt": Deletes the file from the working directory.
 - git status: Shows the file is now unstaged.
 - git ls-files: Lists the files 'Initial-commit.txt' and 'second-commit.txt'.
 - git commit -m "Deletion of Working_with_Branches.txt": Commits the deletion of the file.
 - git status: Final status showing the commit has been made.
- Explorer:** The Explorer sidebar shows two files: 'Initial-commit.txt' and 'second-commit.txt'.
- Outline:** The Outline sidebar shows 'File symbols found in document second-commit.txt'.
- Timeline:** The Timeline sidebar shows a single entry: 'added second-commit.txt'.

Fig. Delete successful from staging area

Undoing Unstaged Changes?

Added Extra Information not needed on Initial -Commit .txt

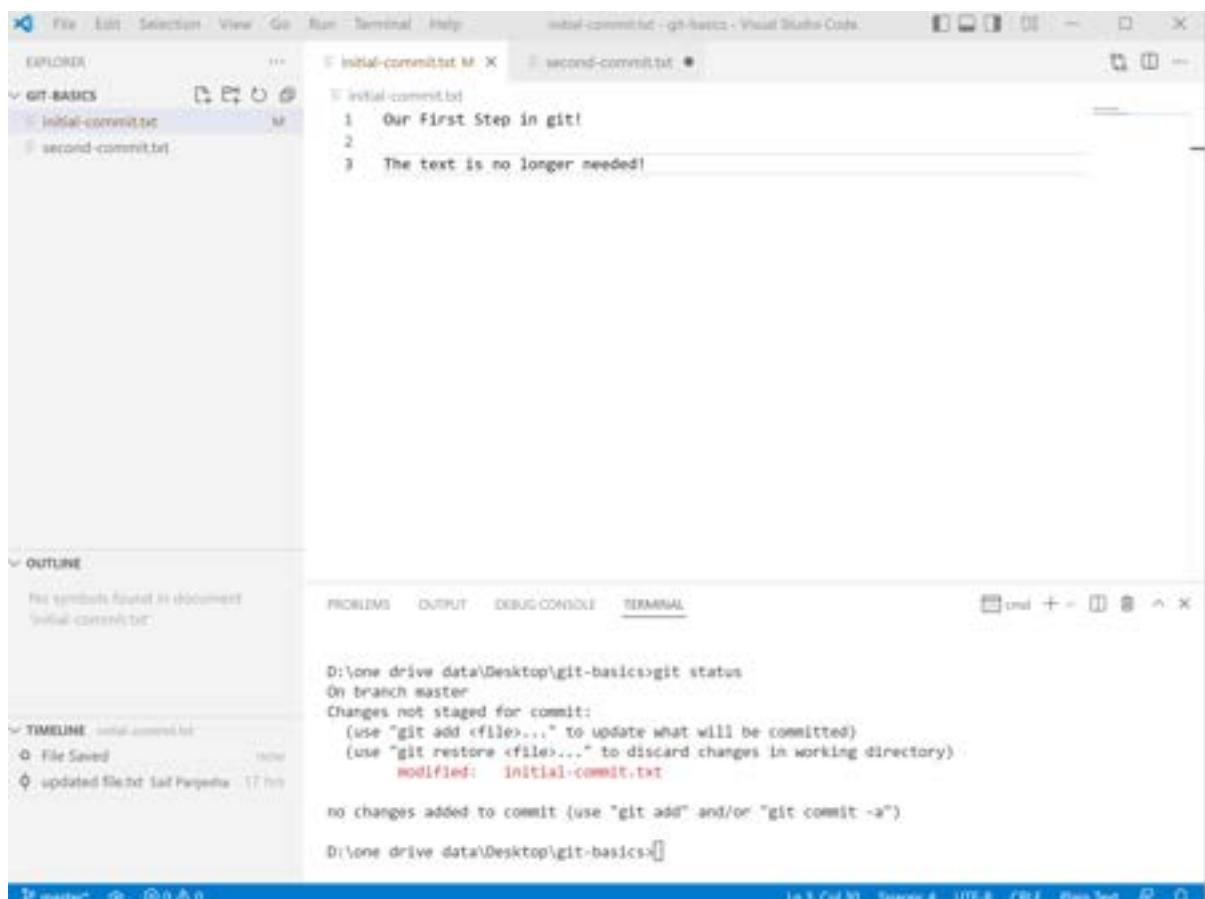


Fig. Unstaged Changes

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

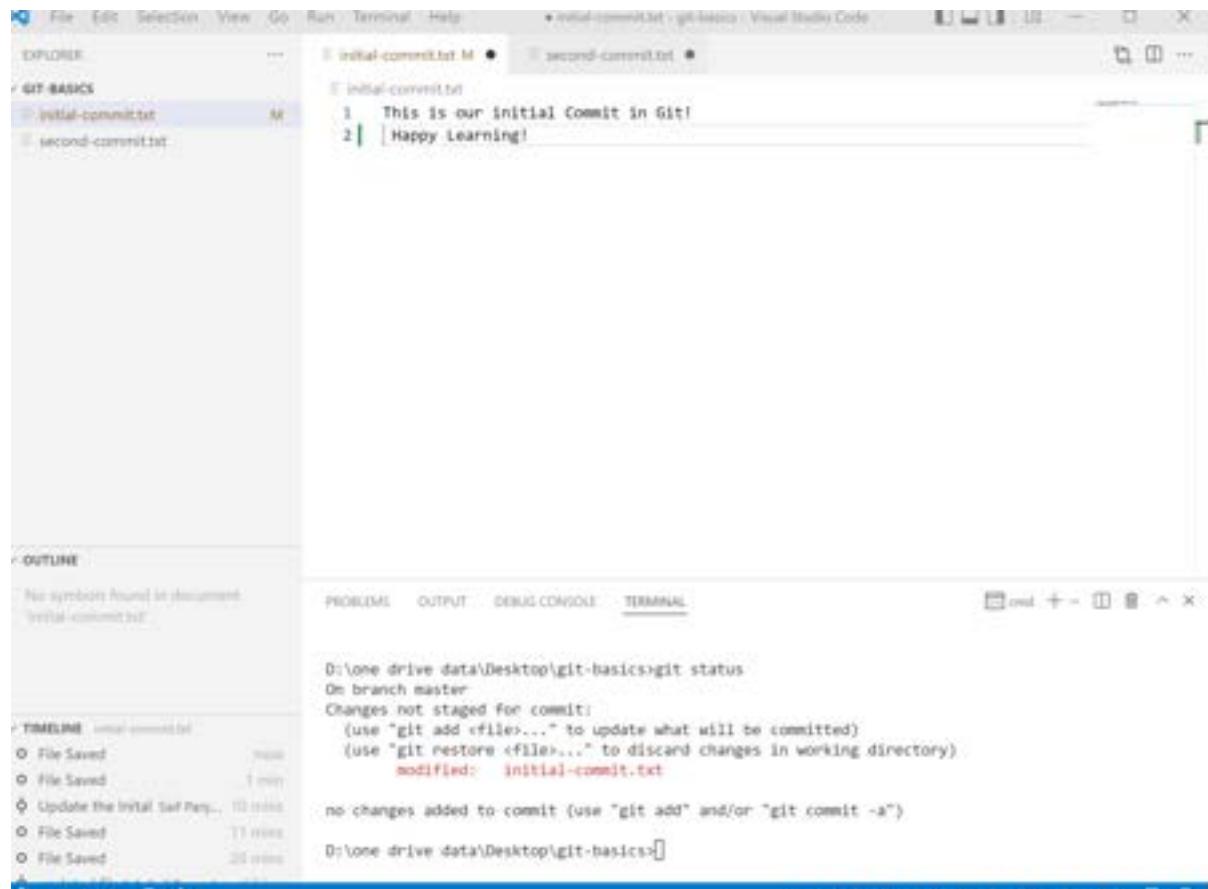
D:\one drive data\Desktop\git-basics>git status
On branch master
nothing to commit, working tree clean

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 commits: 'initial-commit.txt' and 'second-commit.txt'. The 'initial-commit.txt' commit is selected. The terminal tab shows the output of a 'git status' command. The output indicates that the file 'initial-commit.txt' is modified and unstaged. The terminal also shows the command 'git status' being run.

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

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

Fig. Unstaged Changes Happens



The screenshot shows the Visual Studio Code interface with the terminal tab active. The terminal shows the command 'git restore initial-commit.txt' being run. The output of the 'git status' command shows that the working tree is now clean, indicating that the unstaged changes have been restored.

```
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 Explorer:** Shows two files: "initial-commit.txt" and "second-commit.txt".
- Editor:** Displays the content of "initial-commit.txt" which reads:

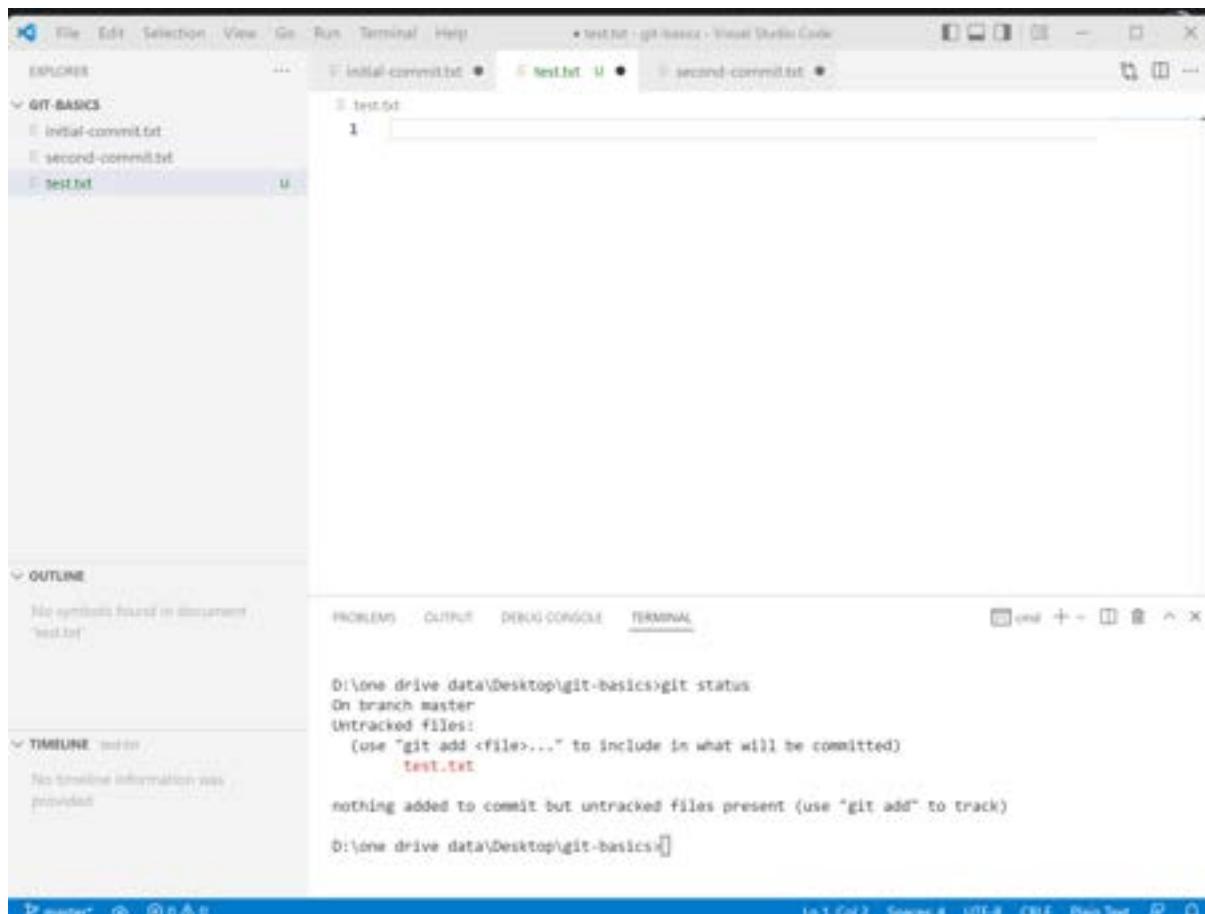
```
1 This is our initial Commit in Git!
```
- Terminal:** Shows the command history and output:

```
D:\One Drive Data\Desktop\git-basics>git restore .
D:\One Drive Data\Desktop\git-basics>git status
On branch master
nothing to commit, working tree clean
D:\One Drive Data\Desktop\git-basics>
```
- Timeline:** Shows a history of file saves and updates for "initial-commit.txt".

Fig. Successful restore back the changes on multiple statement

How to restore Unstaged File:

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



The screenshot shows the Visual Studio Code interface. In the Explorer panel on the left, there are three items: 'initial-commit.txt', 'second-commit.txt', and 'test.txt'. The 'test.txt' item is selected and has a green 'U' icon next to it, indicating it is untracked. In the center workspace, there is a single line of text: '1'. Below the workspace, the Terminal tab is active, showing 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>
```

Fig Created file test.txt



The screenshot shows the Visual Studio Code interface with the same setup as the previous figure. The terminal output now includes the results of running 'git clean' commands:

```
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)

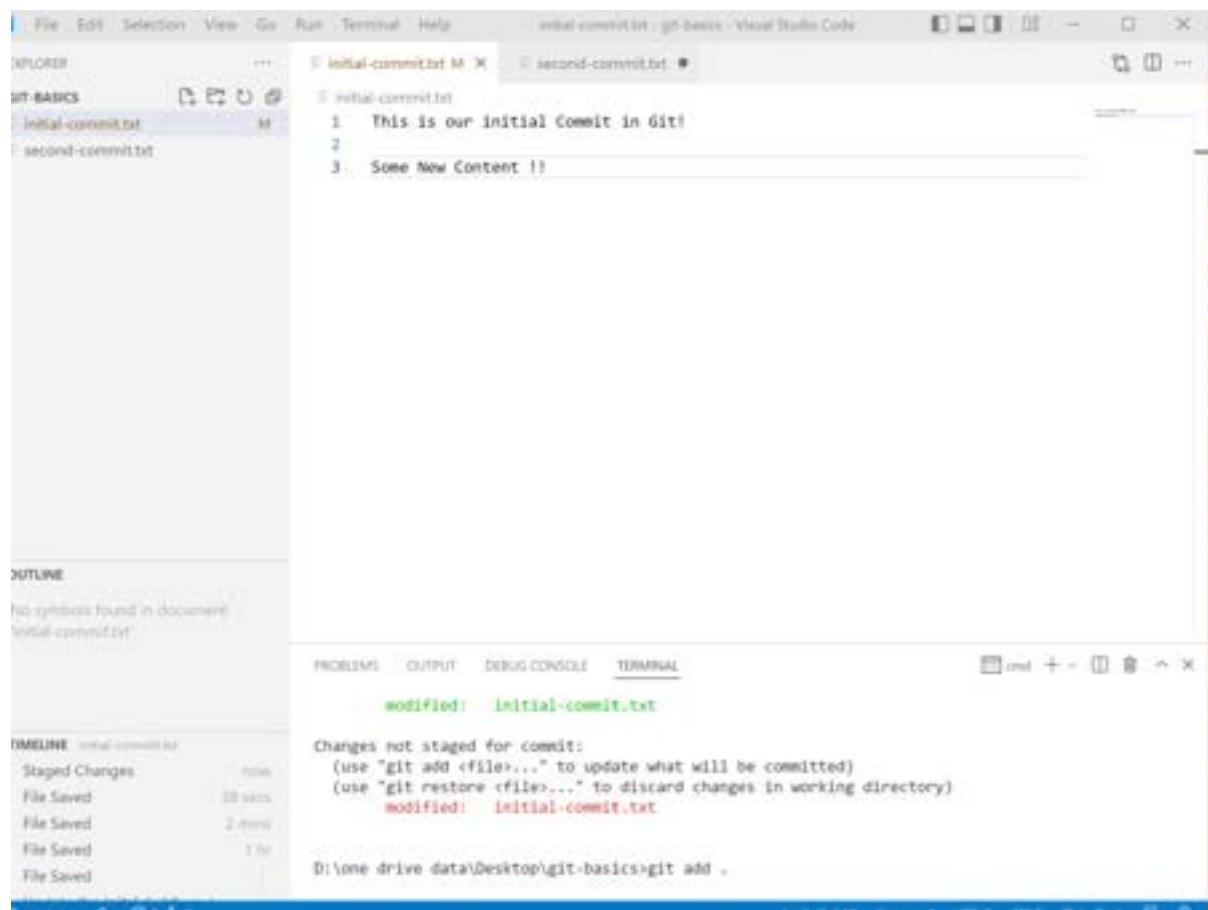
D:\One Drive\DATA\Desktop\git-basics>git clean -dn
Would remove test.txt

D:\One Drive\DATA\Desktop\git-basics>git clean -df
Removing test.txt

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

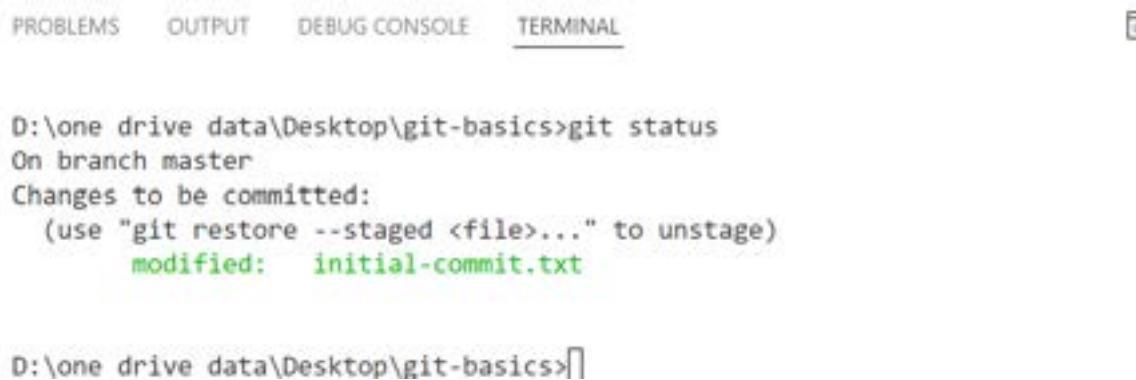
Fig. Remove test.txt

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, showing its contents: '1 This is our initial Commit in Git!', '2', and '3 Some New Content !!'. The terminal tab is active, displaying the command 'git add .' and its output: 'modified: initial-commit.txt'. The status bar at the bottom shows 'D:\one drive data\Desktop\git-basics>git add .'.

Fig. Added Some Content



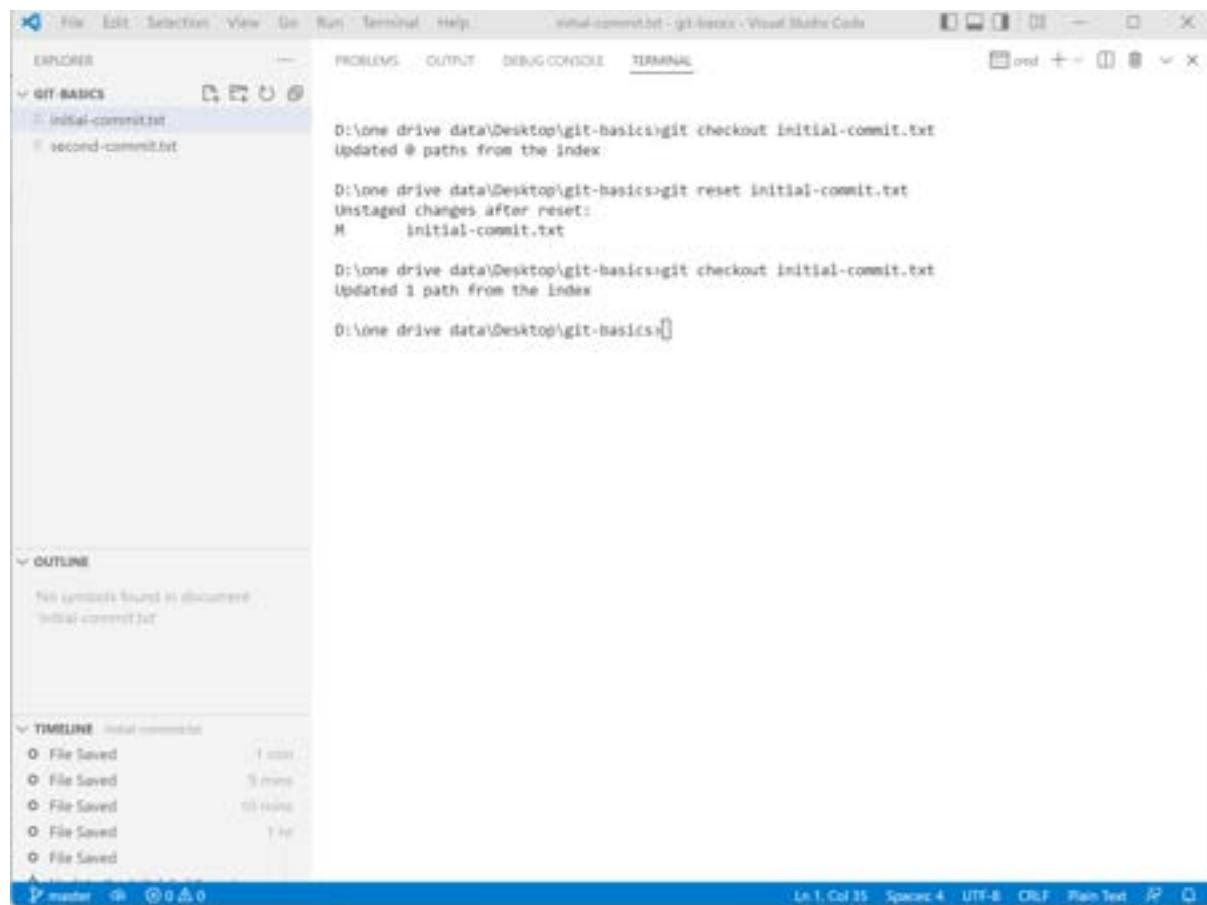
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
```

```
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 a Visual Studio Code interface with the following details:

- EXPLORER**: Shows two files: "initial-commit.txt" and "second-commit.txt".
- TERMINAL**: Displays the following command-line session:

```
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**: Shows "initial-commit.txt" as the current document.
- TIMELINE**: Shows a list of five "File Saved" events for "initial-commit.txt" at different times.

Fig. Staging Changes reset

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

The screenshot shows two separate terminal sessions within Visual Studio Code, both titled "second-commit.txt - git-basics - Visual Studio Code".

Top Terminal Session:

```
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>
```

Bottom Terminal Session:

```
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>
```

The screenshots illustrate the use of `git restore` to manage staged changes. In the top session, the user runs `git status`, `git add .`, and `git status` again, showing the file `second-commit.txt` is modified and staged. In the bottom session, the user runs `git status`, then `git restore --staged second-commit.txt`, and finally `git status` again, which shows the file is no longer staged.

Fig. Success Staged Changes restore

Deleting commit with git reset

The screenshot displays two windows side-by-side. The left window is Visual Studio Code, and the right window is the GitHub desktop application.

Visual Studio Code Terminal Output:

```
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>
```

Github Desktop Timeline:

- Initial Commit: initial-commit.txt
- Second Commit: second-commit.txt
- Unrequired File: unrequired.txt (by Saif Panjesha)

GitHub Desktop Terminal Output:

```
D:\One Drive\DATA\Desktop\git-basics>git log
commit be580d266517e57dd716627f1acfef598fcbb0d3a (HEAD -> master)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Thu May 19 22:20:15 2022 +0530

    unrequired.txt

commit f4fd19c066db84e9582c941d12ae898b3997696d
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Thu May 19 22:09:48 2022 +0530

    Unrequired File

commit 26eccace066f791648def83186b6153289a157ac
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Thu May 19 20:16:58 2022 +0530

    Update the Initial

commit ccf3569cb84dbc98f730a736a8aba7e8ff39d16af
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Thu May 19 18:31:46 2022 +0530
```

Fig. Logs Heading at Master staging area

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

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

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

- commit 26eccace006f701648def83186b6153289a157ac (HEAD -> master)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date: Thu May 19 20:16:58 2022 +0530
Update the Initial
- commit ccf3569c84dbc98f738a736a8aba7e8ff39d16af
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 cd8816cdbe74cechaf4f5995e25b2f9bab3cb9a3 (quickFix, commitStage, Landing-Page)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date: Thu May 19 15:44:16 2022 +0530
Branches are Working

The timeline sidebar on the left shows two entries: "Staged Changes" and "File Saved".

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 dc4467141cb1c3529cdc7da6200e9600f0c56187 (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 320726660068faa062f8e245df7e3be52c78accc
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 'unrequired.txt' file is currently selected. 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:

- EXPLORER**: Shows three files: `initial-commit.txt`, `second-commit.txt`, and `unrequired.txt`.
- TERMINAL**: Displays the command `git reset HEAD~2` 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 dc4a67141cb3c3529cdc7da6290e9600fbc56387 (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 329726669068faa962f8e245df7e3be52c76accc
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Thu May 19 02:36:24 2022 +0530

    updated file.txt

D:\One drive\data\Desktop\git-basics>
```
- OUTLINE**: Shows no symbols found in document `unrequired.txt`.
- TIMELINE**: Shows a single entry: `File Saved` 12 mins ago.

Fig. Success Deletion on unrequired file

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

- EXPLORER**: Shows three files: `initial-commit.txt`, `second-commit.txt`, and `unrequired.txt`. `unrequired.txt` is selected.
- TERMINAL**: Displays the command `git log` and its output:

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

Fig. Added file again

“git reset” with “- -hard” as Hard reset

Removing all changes from Working directory & Staging area

The screenshot shows the Visual Studio Code interface. The Explorer sidebar shows files: initial-commit.txt, second-commit.txt, and unrequired.txt. The unrequired.txt file contains "Some content". The terminal tab is active, displaying the command: D:\One Drive\DATA\Desktop\git-basics>git reset --hard HEAD~1. The output shows: HEAD is now at dc4a671 added second-commit.txt. The terminal then runs git ls-files, showing initial-commit.txt and second-commit.txt. The Timeline sidebar shows a single entry: File Saved 24 mins ago.

Fig. -- hard removing all area

The screenshot shows the Visual Studio Code interface. The Explorer sidebar shows files: initial-commit.txt, second-commit.txt, and unrequired.txt. The terminal tab is active, displaying the command: D:\One Drive\DATA\Desktop\git-basics>git log. The output shows: commit dc4a67141eb3c3529cdc7da6200e9500f0c56387 (HEAD -> master). Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com> Date: Thu May 19 15:02:23 2022 +0530 added second-commit.txt. The terminal then runs git log again, showing the same commit information. The Timeline sidebar shows a single entry: File Saved 24 mins ago.

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

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

The terminal window has a light gray background with black text. The code blocks are color-coded: blue for directory paths, green for file names, and red for error messages. The terminal tab is highlighted in blue at the top of the window.

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, and a new file named Dummy.txt.
- TERMINAL:** Displays the command history:

```
D:\one drive data\Desktop\git-basics>git add .
D:\one drive data\Desktop\git-basics>git log
commit 1E7EB33e1bab418e8659ceb98f68dc7e7e944a4 (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' and 'Dummy.txt'.
- TIMELINE:** Shows 'File Saved' at 1 min ago and 'File Saved' at 2 min 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

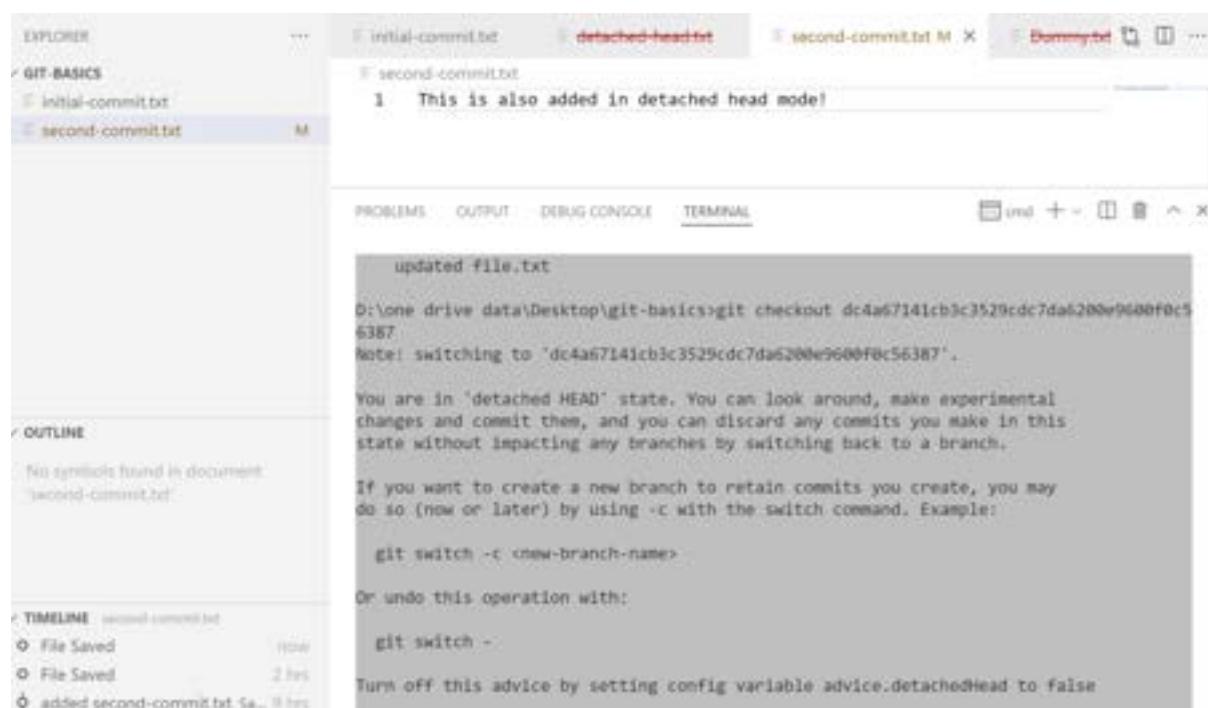


Fig. Added Some in Detached Head

```
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(+)
```

The screenshot shows a Visual Studio Code interface. The terminal window at the bottom displays the following command-line session:

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

The Explorer sidebar on the left shows a folder named "GIT-BASICS" containing files: "detached-head.txt", "dummy.txt", "initial-commit.txt", and "second-commit.txt". The "initial-commit.txt" file is open in the editor, showing its content:

```
1 I Forget this tag Good to have Detached !!
```

The status bar at the bottom indicates "initial-commit.txt : git-basics - Visual Studio Code".

Fig. Succesful head Changes

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

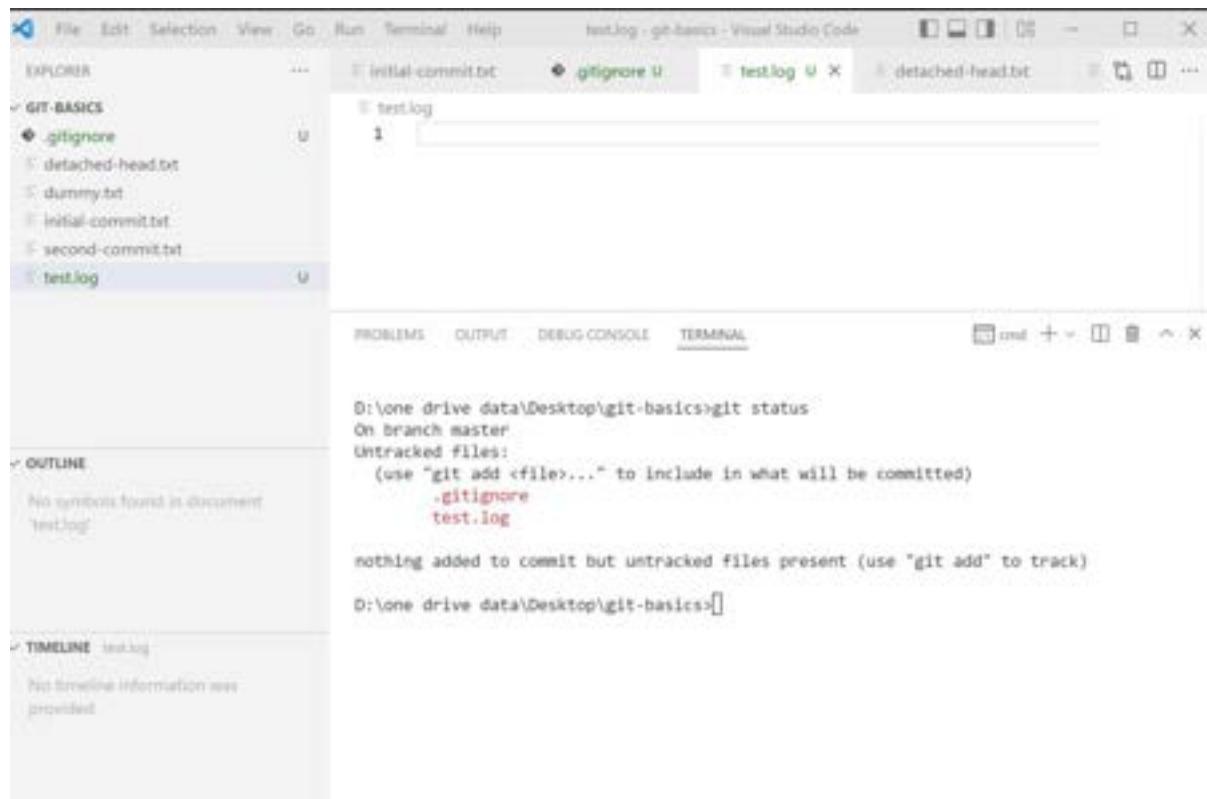
```
Deleted branch Detached2Head (was 4578010).
```

```
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. In the Explorer sidebar, under the 'GIT-BASICS' section, there is a file named '.gitignore'. In the main editor area, there is a file named 'test.log'. The terminal tab shows the command 'git status' being run, which outputs:

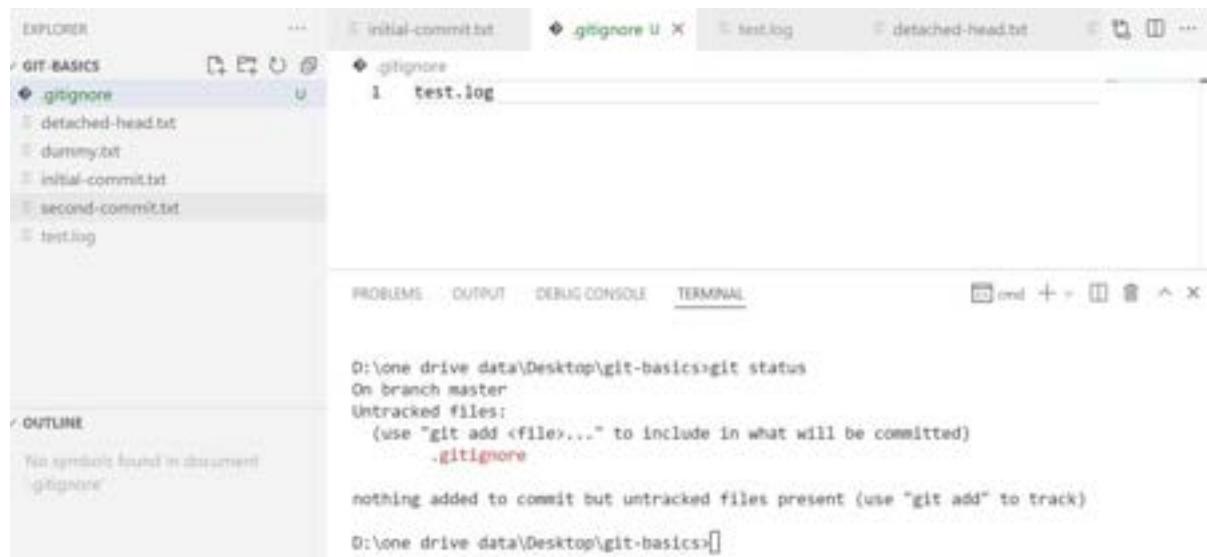
```
D:\One Drive\DATA\Desktop\git-basics>git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .gitignore
    test.log

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

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

Fig. ignore file created

Ignoring test.log File



The screenshot shows the Visual Studio Code interface. In the Explorer sidebar, under the 'GIT-BASICS' section, there is a file named '.gitignore'. In the main editor area, there is a file named 'test.log'. The terminal tab 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

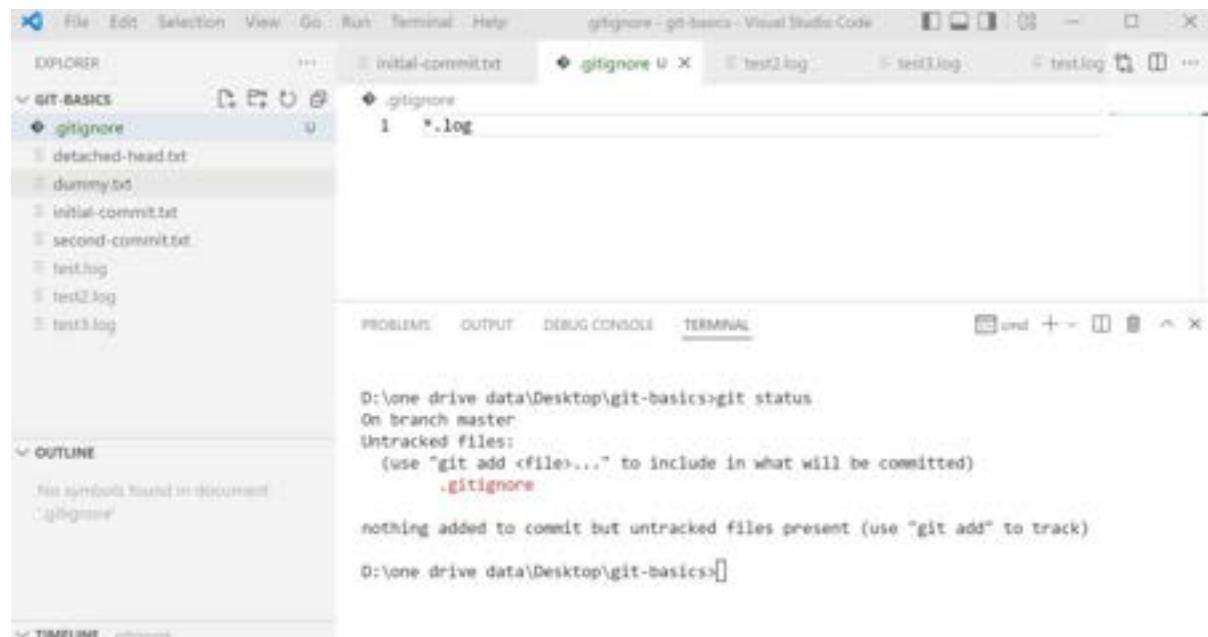
nothing added to commit but untracked files present (use "git add" to track)

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

Fig. test.log File ignored

To Ignore Multiple Files:

***.log – for ignorance all files**



The screenshot shows the Visual Studio Code interface. In the Explorer sidebar, under the 'gitignore' file, there are entries for 'detached-head.txt', 'dummy.txt', 'initial-commit.txt', 'second-commit.txt', 'test.log', 'test2.log', and 'test3.log'. The 'gitignore' file itself contains the following content:

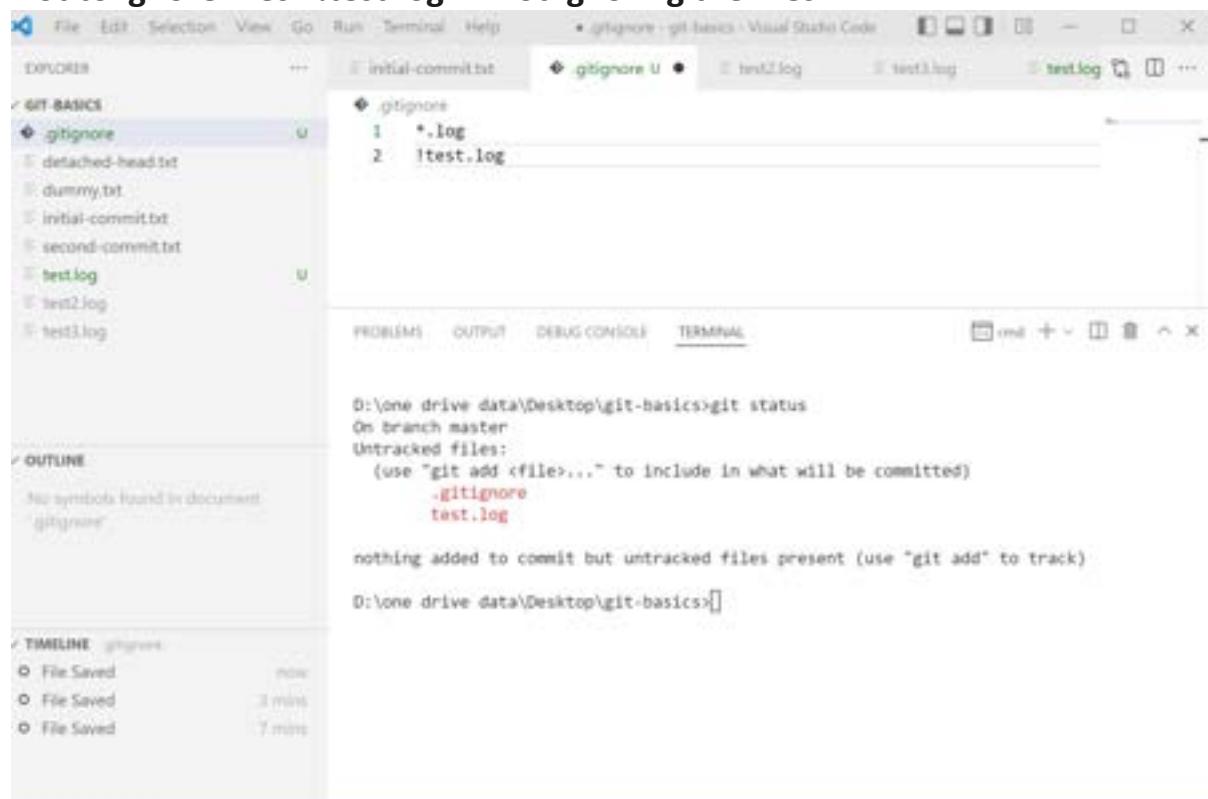
```
1 *.log
```

In the Terminal tab, the command 'git status' is run, showing the following 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
    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. In the Explorer sidebar, under the 'gitignore' file, there are entries for 'detached-head.txt', 'dummy.txt', 'initial-commit.txt', 'second-commit.txt', 'test.log', 'test2.log', and 'test3.log'. The 'gitignore' file itself contains the following content:

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

In the Terminal tab, the command 'git status' is run, showing the following 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

```
File Edit Selection View Go Run Terminal Help
gtignore - git-basics - Visual Studio Code
EXPLORER ... initial-commit.txt .gitignore index.html test2.log test3.log
GIT-BASICS D+ D- ⌂ ⓘ
app-data ⓘ index.html ⓘ
gitignore ⓘ
detached-head.txt ⓘ
dummy.txt ⓘ
initial-commit.txt ⓘ
second-commit.txt ⓘ
test.log ⓘ
test2.log ⓘ
test3.log ⓘ
OUTLINE
No symbols found in document.
gitignore

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
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

```
File Edit Selection View Go Run Terminal Help
gtignore - git-basics - Visual Studio Code
EXPLORER ... initial-commit.txt .gitignore index.html test2.log test3.log
GIT-BASICS D+ D- ⌂ ⓘ
app-data ⓘ index.html ⓘ
gitignore ⓘ
detached-head.txt ⓘ
dummy.txt ⓘ
initial-commit.txt ⓘ
second-commit.txt ⓘ
test.log ⓘ
test2.log ⓘ
test3.log ⓘ
OUTLINE
No symbols found in document.
gitignore

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
D:\One Drive Data\Desktop\git-basics>git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .gitignore
    app-data/
    test.log

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

D:\One Drive Data\Desktop\git-basics>[]
```

Fig. Not ignored folders without “/” command

Clean all Files & Folders: Deletes untracked Files

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

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

```
*.log
!test.log
```

In the bottom right corner, the Terminal tab is active, displaying the command and its output:

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

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

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

Fig. Clean all files & folders

Wrap Up What We Learn:

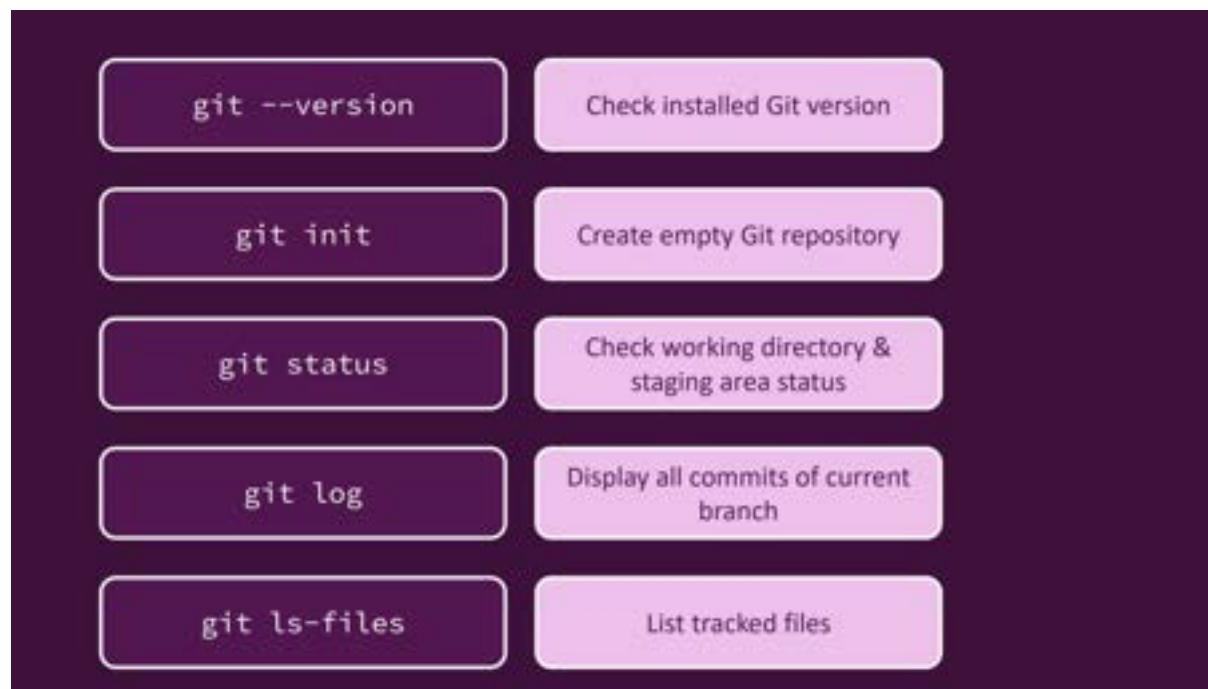


Fig . General Commands



Fig. Basic Commit Creation and access



Fig. Branch and creation access

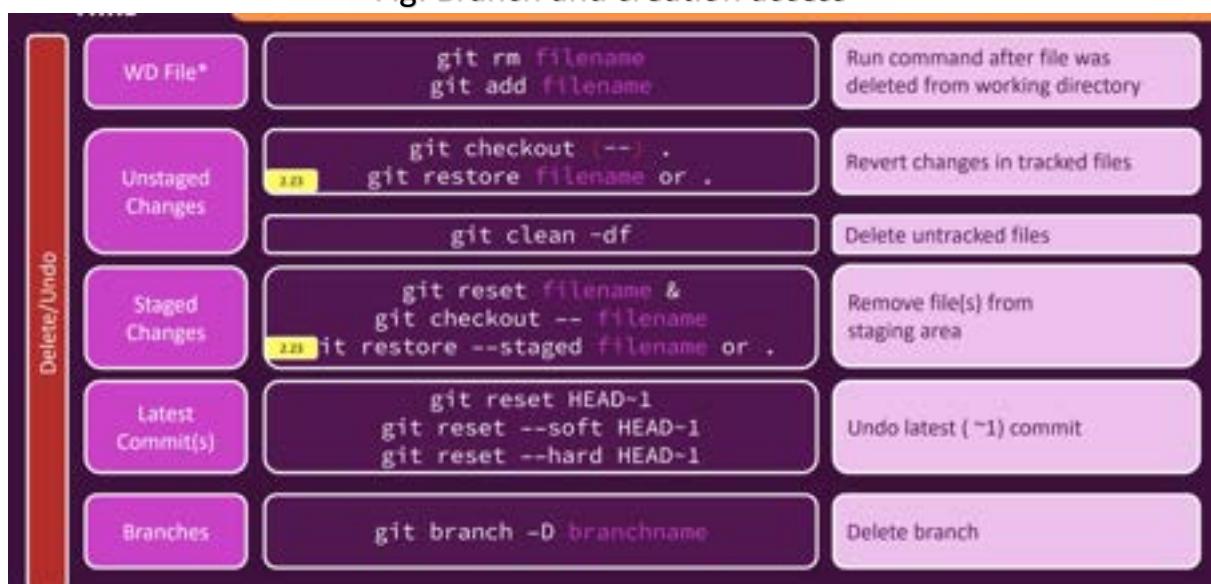


Fig. Deleting Data Summary

Diving Deep into Git

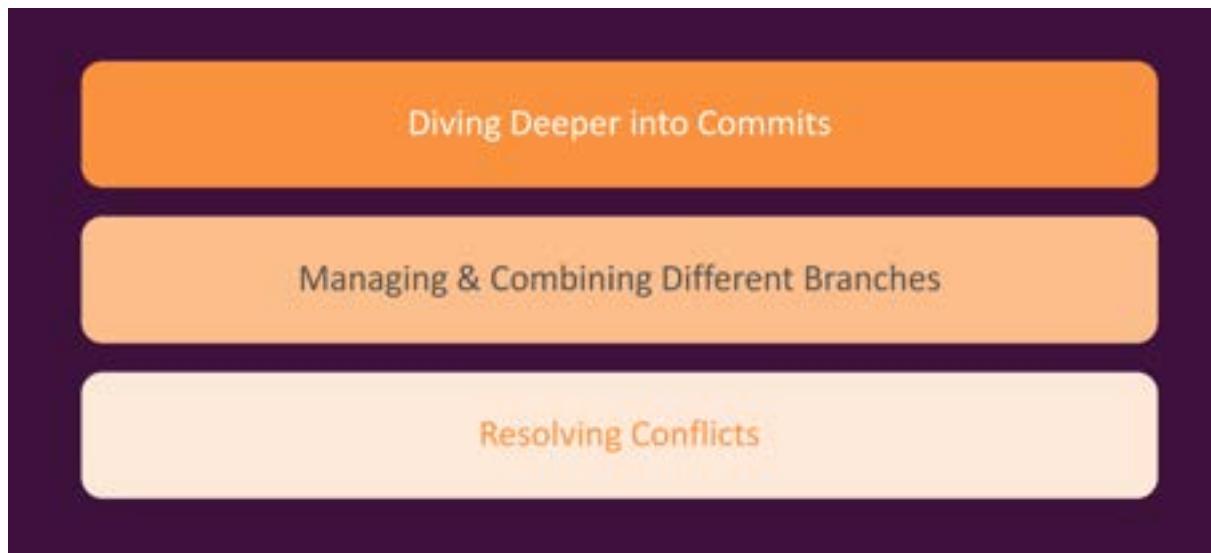


Fig. Module Introduction

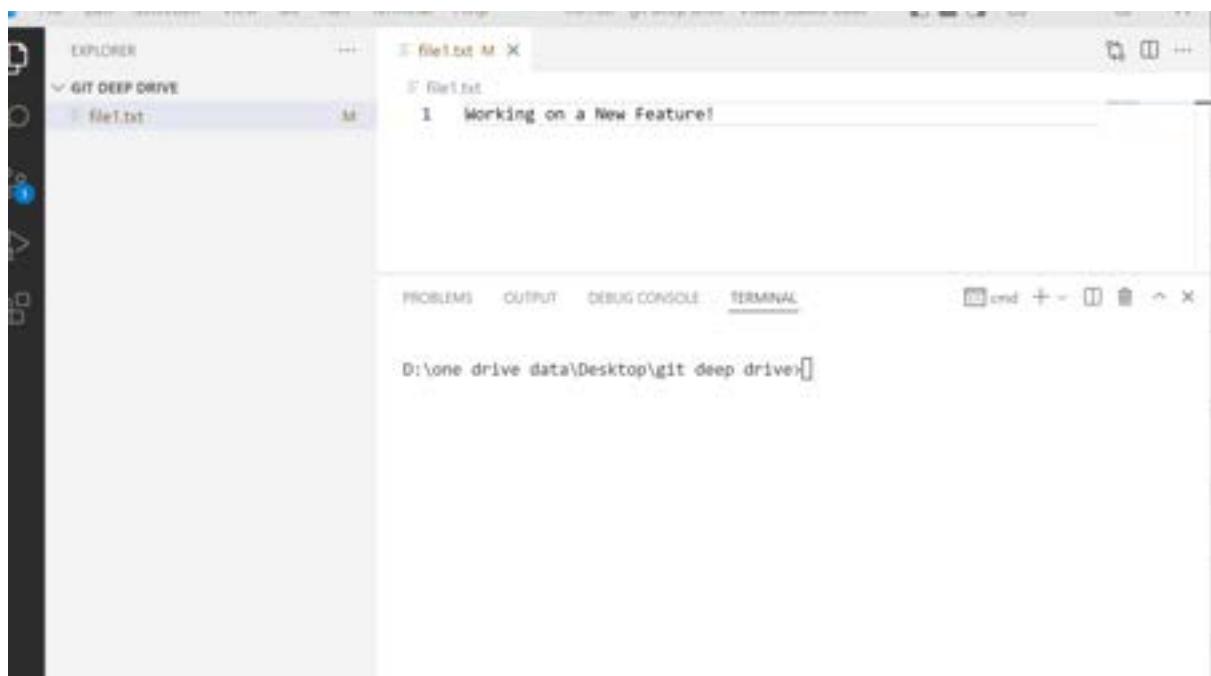
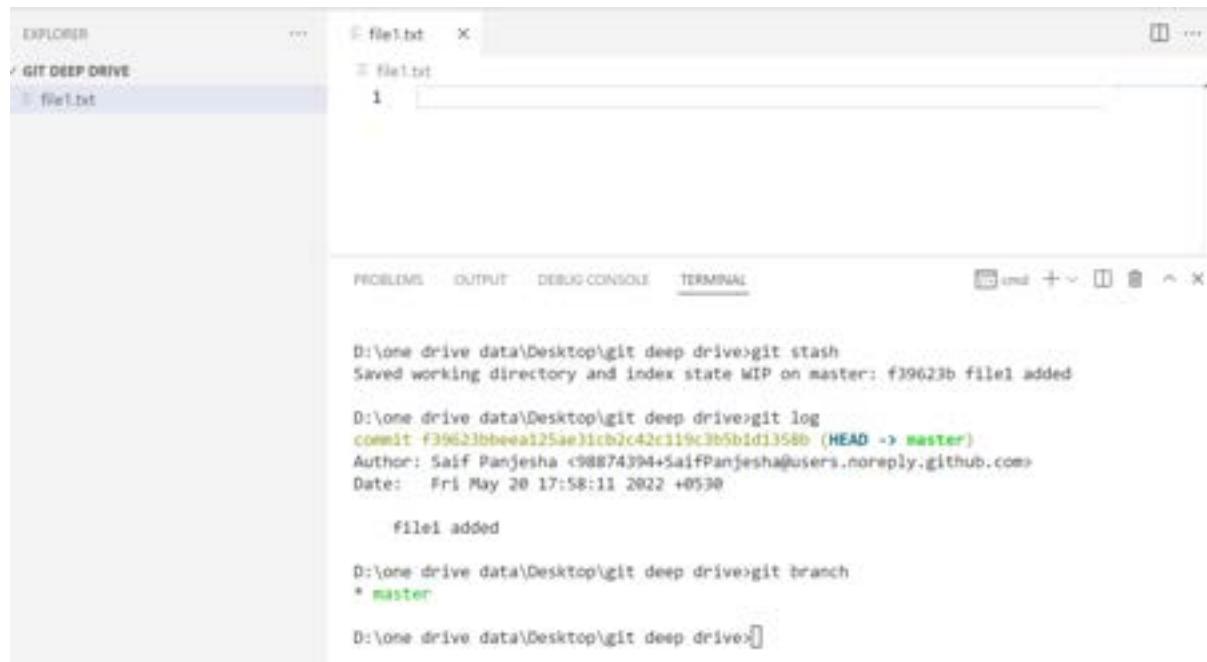


Fig. Created new file into Git Deep Drive Folder

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

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



```
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 f39623b6ea125ae311b2c42e119c3b5b1d1358b (HEAD -> master)
Author: Saif Panjesta <98874394+SaifPanjesta@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



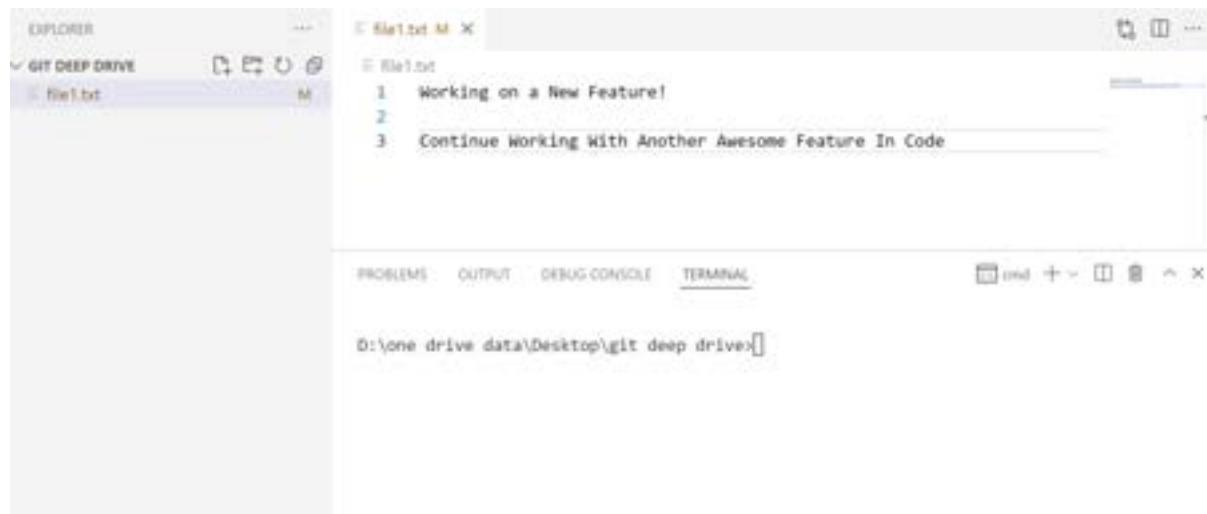
```
D:\one drive data\Desktop\git deep drive>git stash apply
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
    (use "git restore <file>..." to discard changes in working directory)
      modified:   file1.txt

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 'TERMINAL' tab at the bottom shows the command line path 'D:\one drive data\Desktop\git deep drive\[]'. The main editor area displays the contents of 'file1.txt':

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

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 'TERMINAL' tab at the bottom shows the command line path 'D:\one drive data\Desktop\git deep drive\[]'. The terminal output shows the results of running 'git stash':

```
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\[]
```

Fig. git stash Work Saved

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

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

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

D:\One Drive Data\Desktop\git deep drive>
```

Fig. git stash apply

git stash list: List of all Stash Changes

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

```
D:\One Drive Data\Desktop\git deep drive>git stash 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

file1.txt M X

file1.txt

1 Working on a New Feature!

2

3 Continue Working With Another Awesome Feature In Code

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

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

file1.txt M X

file1.txt

1

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

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

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

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

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

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

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

stash@{3}: WIP on master: f39623b file1 added

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

Fig. git stash changes saved in working directory

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

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

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

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

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

D:\One Drive\DATA\Desktop\git deep drive>
```

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

To check our latest changes:

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

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

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

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

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

Fig. To Check our Latest changes with git stash apply

We can add message to our stash:



Fig. Third Feature added

stash push -m "Third Feature added"

```
0:\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

0:\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

0:\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: 'Saved working directory and index state On master: Third Feature added'. Below this, the command 'git stash list' is shown, listing six stashes. The first stash is identified as 'On master: Third Feature added'. The subsequent five stashes are labeled as 'WIP on master' with the commit hash 'f39623b' and the file 'file1' added.

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

Now this message we have to add in our Projects:

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

```
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} (a4361775d2171a3bad35e9ac256dd503d4c2c51b)

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

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

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

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



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

  feture 3 added to project

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

  file1 added

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

Fig. Added Successful to project and out from stash stack

git stash clear: Remove the git stash

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL cmd +

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

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

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

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

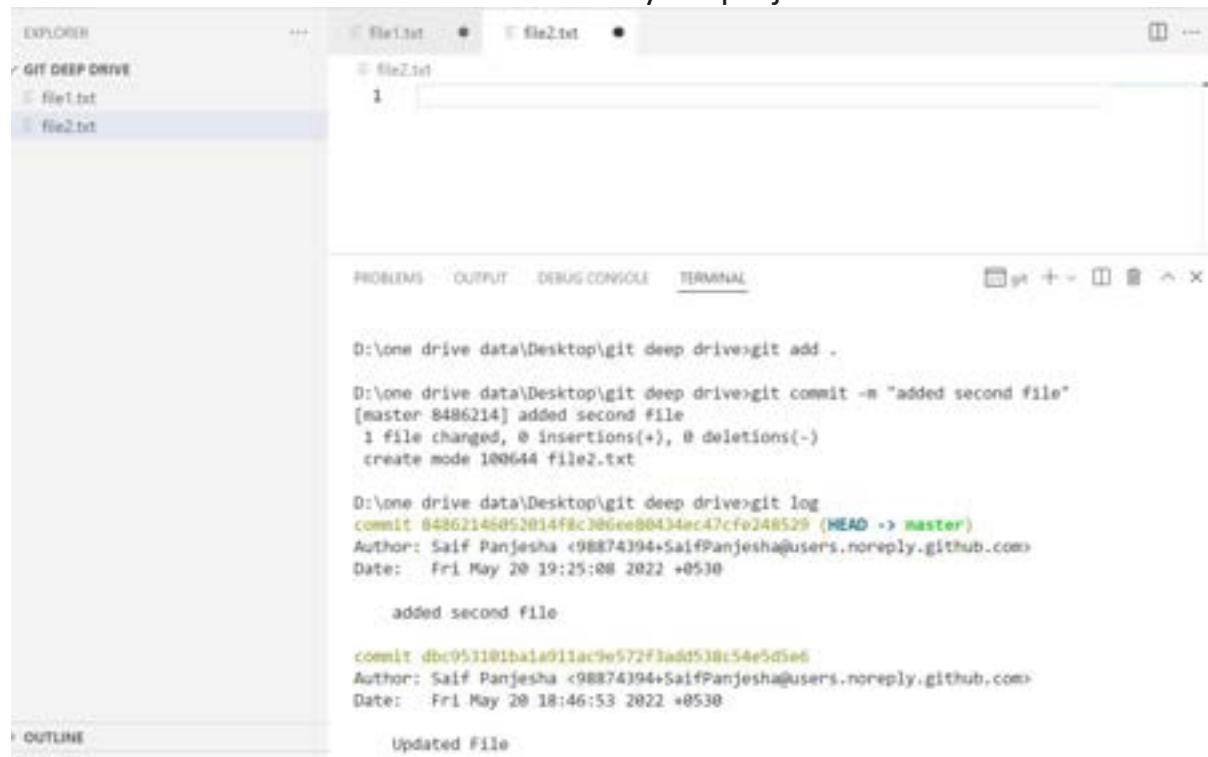
D:\one drive data\Desktop\git deep drive>git stash list

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

Fig. Remove the stash

Bringing the Lost data with “git reflog”

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



The screenshot shows the VS Code interface with the Explorer sidebar open, displaying a folder named "GIT DEEP DRIVE" containing two files: "file1.txt" and "file2.txt". The "file2.txt" tab is active, showing its content: "1". Below the editor, the Terminal tab is selected, showing the command history:

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

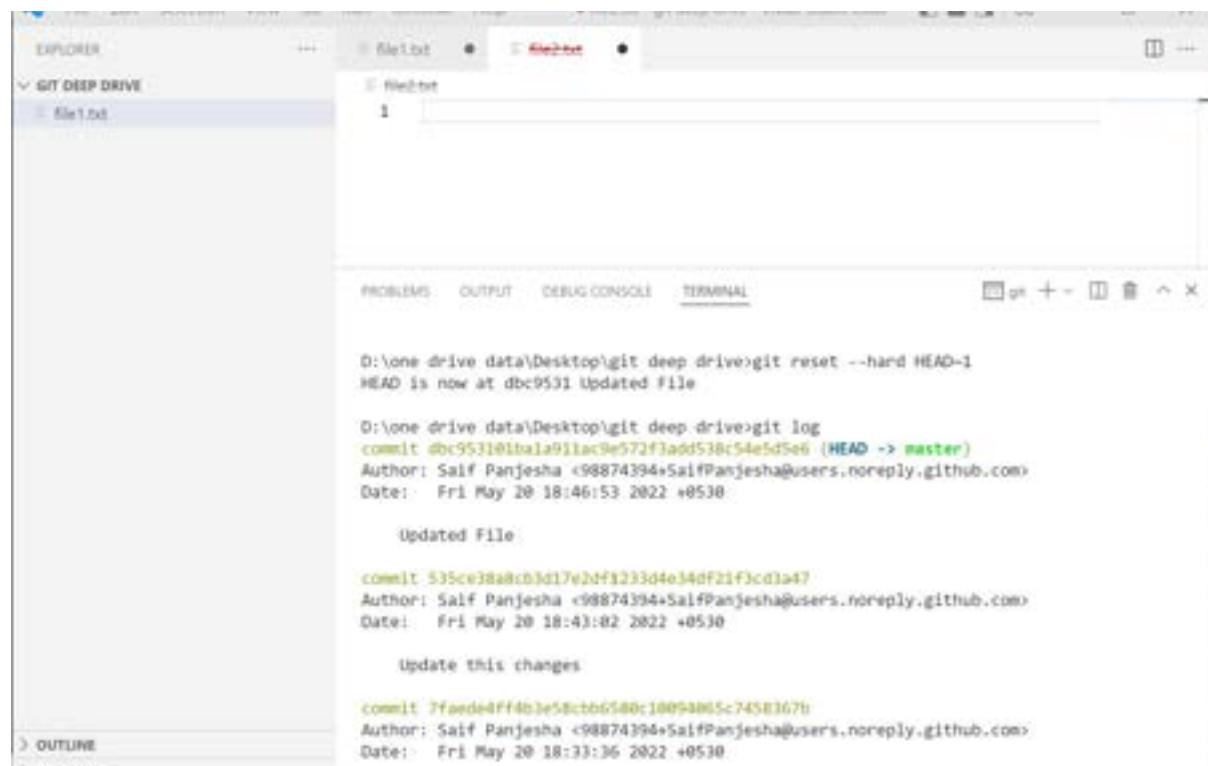
D:\One Drive\DATA\Desktop\git deep drive>git log
commit 84862146052014f8c306ee80434ec47cfe248529 (HEAD -> master)
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date:   Fri May 20 19:25:08 2022 +0530

    added second file

commit dbc9531@lbalal9llac9e572f3add538c54e5d5e6
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date:   Fri May 20 18:46:53 2022 +0530
```

The status bar at the bottom 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 one file: "file1.txt". The "file2.txt" tab is active, showing its content: "1". Below the editor, the Terminal tab is selected, showing the command history:

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

D:\One Drive\DATA\Desktop\git deep drive>git log
commit dbc9531@lbalal9llac9e572f3add538c54e5d5e6 (HEAD -> master)
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date:   Fri May 20 18:46:53 2022 +0530

    Updated File

commit 535ce38a8cb3d17e2df1233d4e34df21f3cd3a47
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date:   Fri May 20 18:41:02 2022 +0530

    Update this changes

commit 7faedea4ff4b3e58ctb6658010093805c7458367b
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date:   Fri May 20 18:33:36 2022 +0530
```

Fig. removed file not in used

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

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

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

```
D:\One Drive\DATA\Desktop\GIT DEEP DRIVE>git reflog
dbc9531 (HEAD -> master) HEAD@{0}: reset: moving to HEAD-1
8486214 HEAD@{1}: reset: moving to 8486214
dbc9531 (HEAD -> master) HEAD@{2}: reset: moving to dbc9531
dbc9531 (HEAD -> master) HEAD@{3}: reset: moving to HEAD-1
8486214 HEAD@{4}: commit: added second file
dbc9531 (HEAD -> master) HEAD@{5}: checkout: moving from 2f9b1aeb37fb28d9ee21cdd72aa47
1c86fb65723 to master
2f9b1ae HEAD@{6}: commit: file 2 added
dbc9531 (HEAD -> master) HEAD@{7}: checkout: moving from master to dbc9531
dbc9531 (HEAD -> master) HEAD@{8}: commit: Updated File
535ce38 HEAD@{9}: commit: Update this changes
7fae0e4 HEAD@{10}: commit: future 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

```
D:\One Drive\DATA\Desktop\GIT DEEP DRIVE>git log
commit 84862146052014fb306ed88434ec47cfef348529 (HEAD -> master)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 19:25:08 2022 +0530

    added second file

commit dbc953101ba911ac9e572f3add518c54e5d5e6
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 18:46:53 2022 +0530

    Updated File

commit 535ce38a8cb1d17e2df1233d4e34df21f3cd3a47
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 18:43:02 2022 +0530

    Update this changes

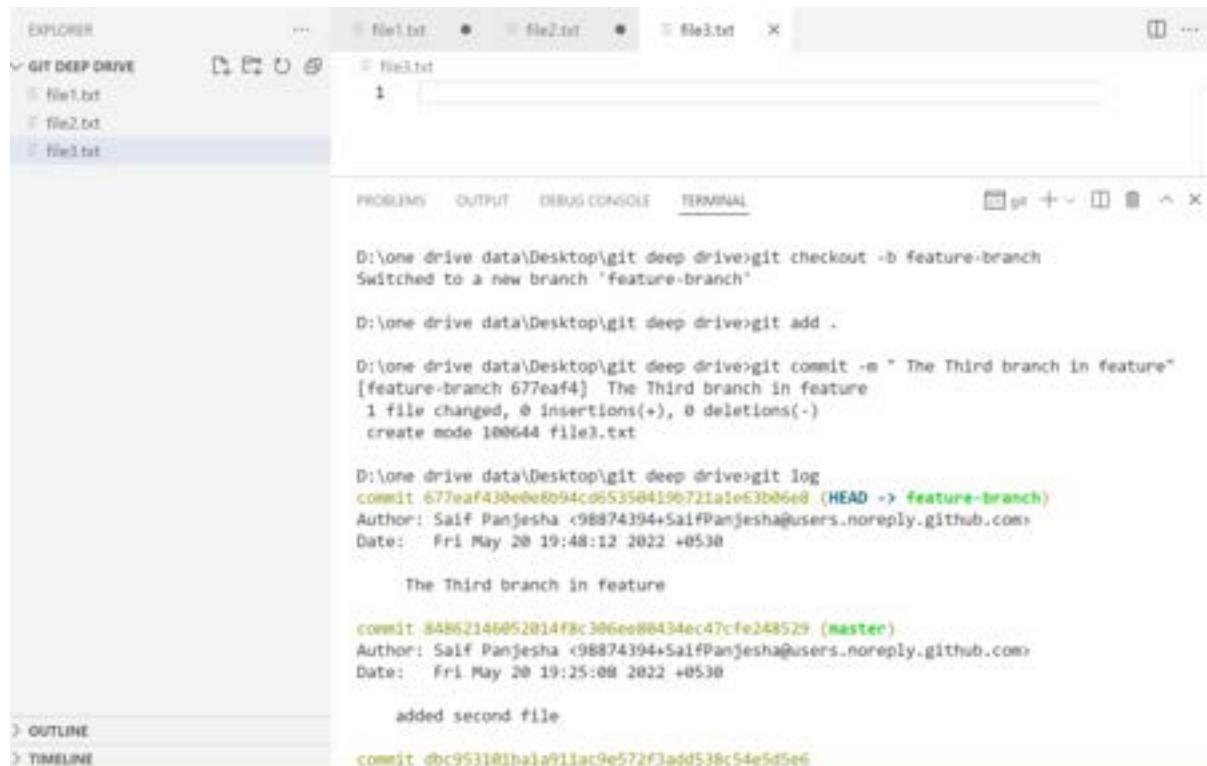
commit 7fae0e4ff4h1e58cb6580c10094065c7458367b
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 18:33:36 2022 +0530

    future 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 terminal tab selected. The terminal output is as follows:

```
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 677eaf430ee6b94cd65358419071iale63b06ed (HEAD -> feature-branch)
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date:   Fri May 20 19:48:12 2022 +0530

  The Third branch in feature

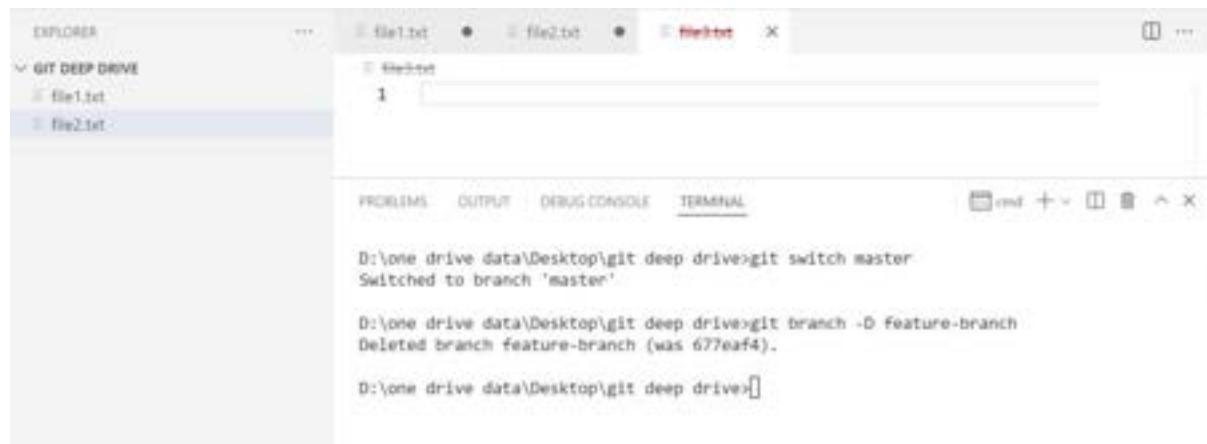
commit 84862146052014f8c306ee88434ec47cfe248529 (master)
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date:   Fri May 20 19:25:08 2022 +0530

  added second file

commit dbc95310ihala9llac9e572f3add538c54e5d5e6
```

Fig. created a new branch with file3.txt

Switch back to master branch:



The screenshot shows the VS Code interface with the terminal tab selected. The terminal output is as follows:

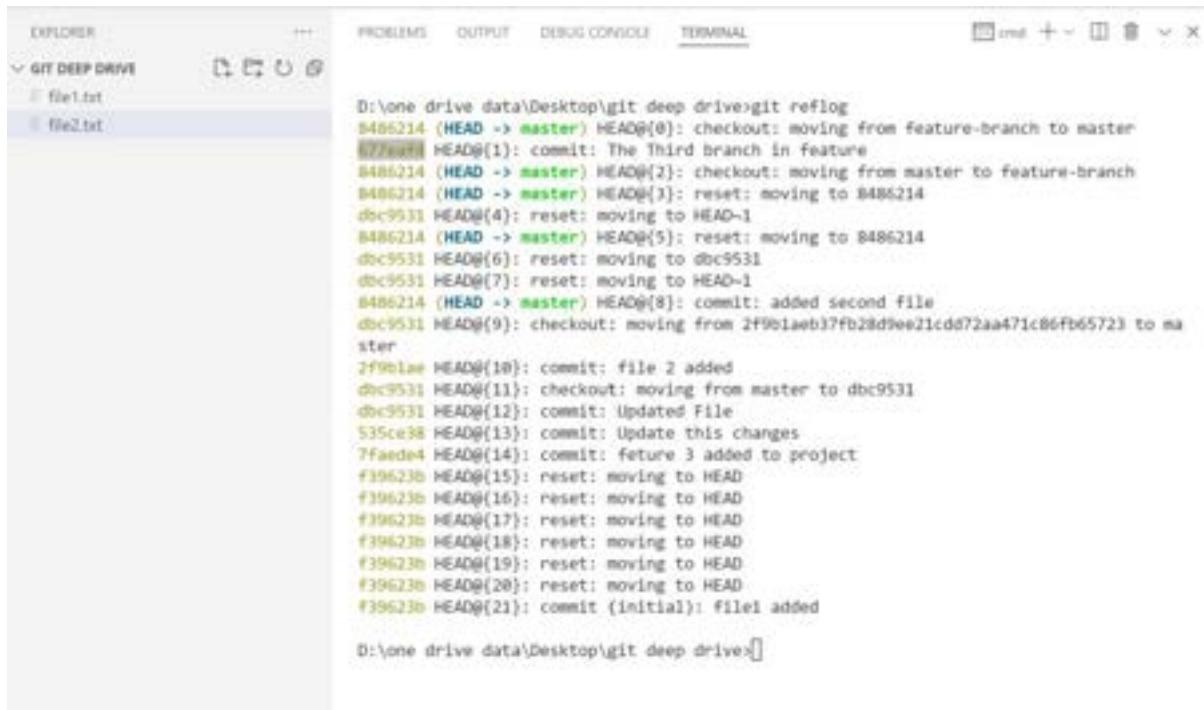
```
D:\One Drive\DATA\Desktop\git\deep\drive>git switch master
Switched to branch 'master'

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
677eaaf 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 2f981aeb37fb28d9ee21cd72aa471c86fb65723 to master
7f9b1ae 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
7fae0e4 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 677eaaf
Note: switching to '677eaaf'.

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 677eaaf The Third branch in feature
D:\one drive data\Desktop\git deep drive>

Fig. updated track ID has added to detached head

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

```
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>
```

Fig. successful added branches with “reflog”

The screenshot shows the VS Code interface with the terminal tab selected. The terminal window displays the following commit history for the 'feature_branch' branch:

```
D:\One Drive\data\Desktop\git deep drive>git log
commit 677eaf430e0e8b94cd65350419b721a1e63b06e8 (HEAD -> feature_branch)
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date:   Fri May 20 19:48:12 2022 +0530

The Third branch in feature

commit 84862146052014fb3086e480434ec47cf248529 (master)
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date:   Fri May 20 19:25:08 2022 +0530

    added second file

commit dbc953181bala91lac9e572f3add538c54e5d5e6
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date:   Fri May 20 18:46:53 2022 +0530

    Updated File

commit 535ce3ba8cb3d17e2df1233d4e34df21f3cd3a47
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date:   Fri May 20 18:43:02 2022 +0530

    Update this changes

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

Fig. Commit history of current branch(feature_branch)

Combining Branches – What & Why?

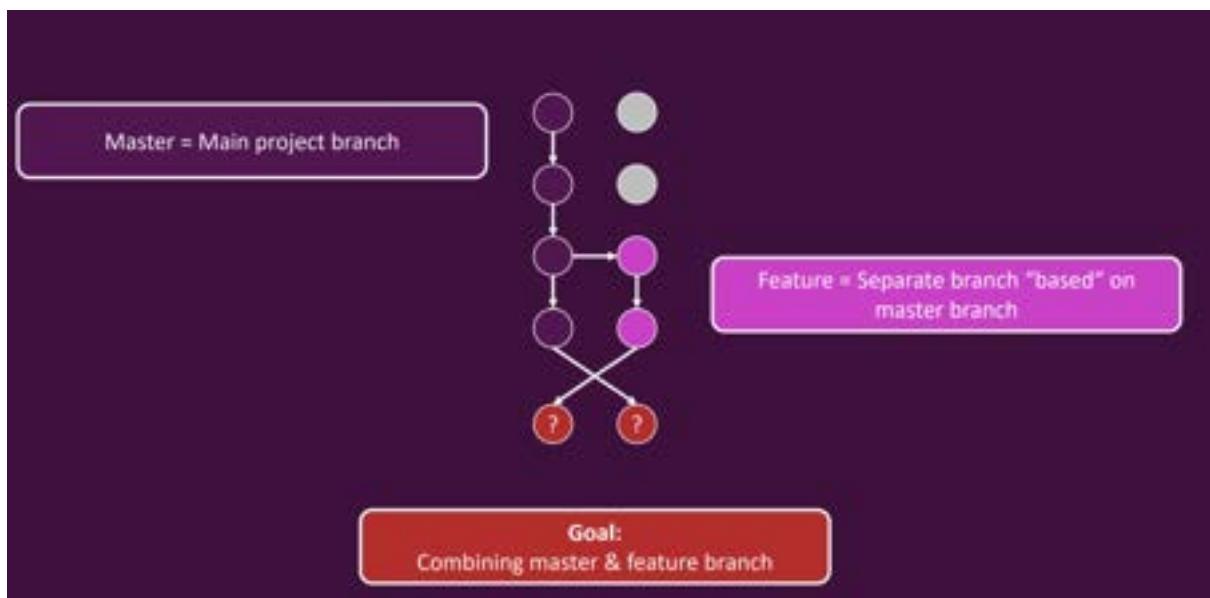


Fig. show Main Branch as Master and Separate Branch as feature

Understanding Merge Types:

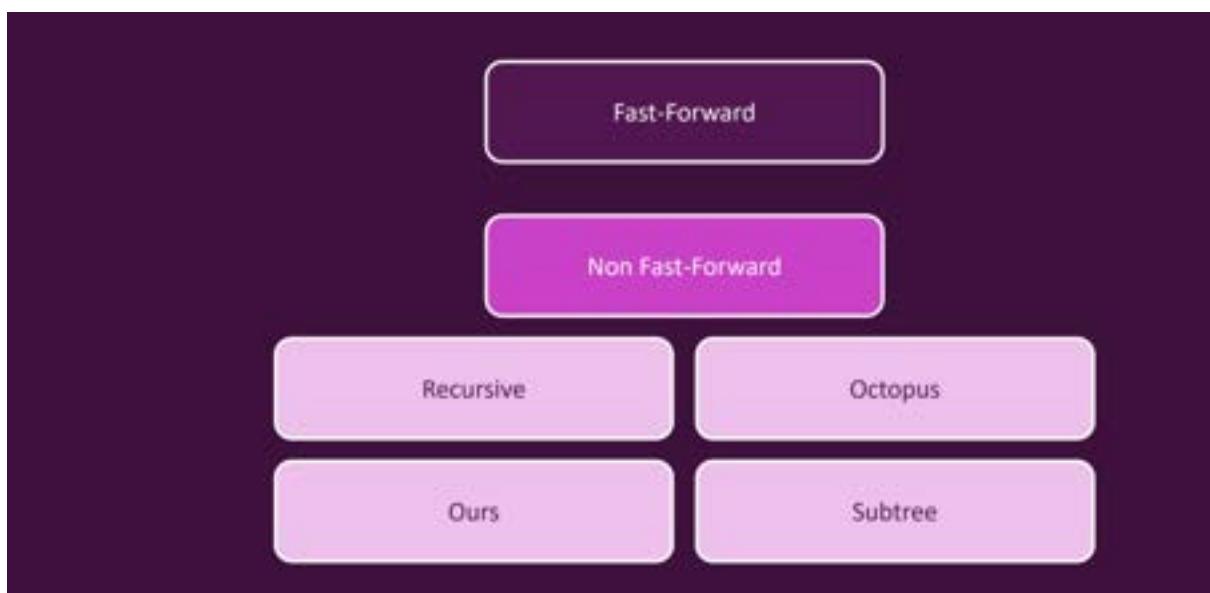


Fig. Shows Types of Merges in Git

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.

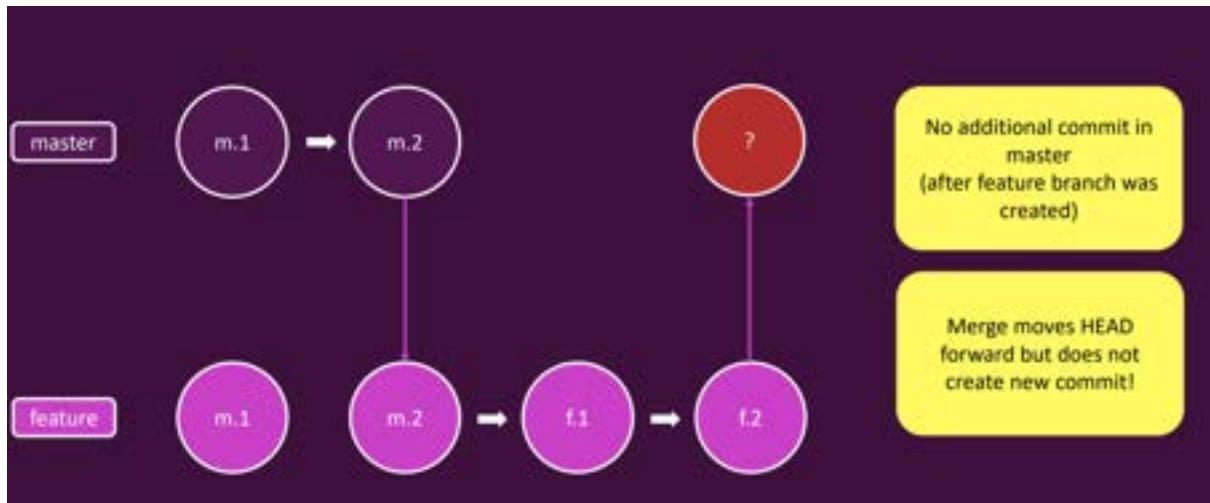


Fig. Shows Master & Feature -Merge("Fast-Forward")

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

```
EXPLORER          ⌂ m1.txt X
                  ⌂ m1.txt
                  master > ⌂ m1.txt
                  1

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   cmd + - ⌂ ^ x

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 262d300f2eaed871e02d14db2de741362f25bdb4 (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 m1.txt second commit

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\Branches>git add .
D:\one\drive\data\Desktop\Branches>git commit -m "m2 added"
[master bdd6aa9] m2 added
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Master/m2.txt

D:\one\drive\data\Desktop\Branches>git log
commit bdd6aa9aae81de33d489d5cedabfa91aec0c2 (HEAD -> master)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:18:20 2022 +0530

  m2 added

commit 262d300f2eaed0871e002d14db2de741382f25bd04
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 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 terminal tab selected. The terminal window displays the following git commands and their output:

```
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 b9d84fcacf79eba1ce29224f1cf141fba98cef34 (HEAD -> feature)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:22:19 2022 +0530

  f2 added

commit b6fc683a2918b3f8nid91984260888192c4df43
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:21:42 2022 +0530

  f1 added
```

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

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

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

D:\one drive data\Desktop\Branches>git log
commit 09d84fcacf79ebabce29224f1cf141fbba98cefaf34 (HEAD -> feature)
Author: Saif Panjesha <98874394+saifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:22:19 2022 +0530

  f2 added

commit b6fc603a2918b3f0b1da9198426b888192c4df43
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 09fac428c3a0b18850aec1c2f611898d0e42ab706
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-line session:

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

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

Fig. Switched to master branch

Fast forward merge happens:

D:\One Drive Data\Desktop\Branches>git switch master
Switched to branch 'master'
D:\One Drive Data\Desktop\Branches>git merge feature
Updating f32b646..89d84fc
Fast-forward
 feature/f1.txt | 0
 feature/f2.txt | 0
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 feature/f1.txt
 create mode 100644 feature/f2.txt
D:\One Drive Data\Desktop\Branches>

Fig. shows merged with master branch as fast - forward

D:\One Drive Data\Desktop\Branches>git log
commit 89d84fcacf79eba8ce29224f1cf1a1fb98cef934 (HEAD -> master, feature)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date: Fri May 20 23:22:19 2022 +0530
 f2 added

commit b6fc683a2918b3f8b1da9198426b888192c4df43
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date: Fri May 20 23:21:42 2022 +0530
 f1 added

commit f12b9469d954ae7edb2082b9f14435494697f398
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date: Fri May 20 23:17:14 2022 +0530
 m2 added

commit 09ac428c3a0b18858aec1c2f011898d8e42ab706
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date: Fri May 20 23:16:24 2022 +0530
 m1 added

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

```
D:\One Drive Data\Desktop\Branches>git reset --hard HEAD~2
HEAD is now at f32b946 m2 added

D:\One Drive Data\Desktop\Branches>git log
commit f32b9469d954ae7e4b2082b9f14435494697f398 <HEAD -> master>
Author: Saif Panjesta <58874394+SaifPanjesta@users.noreply.github.com>
Date:   Fri May 20 23:17:14 2022 +0530

    m2 added

commit 09ac428c3a0f18850aec1c2f613898d0e42ab706
Author: Saif Panjesta <58874394+SaifPanjesta@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

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 f32b9469d954ae7e4b2082b9f14435494697f300 (HEAD -> master)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date: Fri May 20 23:17:14 2022 +0530

m2 added

commit 09ac428c3a0b1885baec1c2f611898d0e42ab706
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>

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 00becf010922522dd53b7735850e92eb474dc11 (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 f32b9469d954ae7e4b2082b9f14435494697f300
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date: Fri May 20 23:17:14 2022 +0530

m2 added

commit 09ac428c3a0b1885baec1c2f611898d0e42ab706
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:

D:\one\drive\data\Desktop\Branches>git log
commit @0bcfc01b922522d053b7735859e02eb4746c31 (HEAD -> master)
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date: Sat May 21 00:09:37 2022 +0530

together master and feature

commit F32b9469d954aa7e4b208219f14415494697f308
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date: Fri May 20 23:17:14 2022 +0530

m2 added

commit @9ac428c1a8b18858aec1c2f611898d0e42ab706
Author: Saif Panjesta <98874394+SaifPanjesta@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):

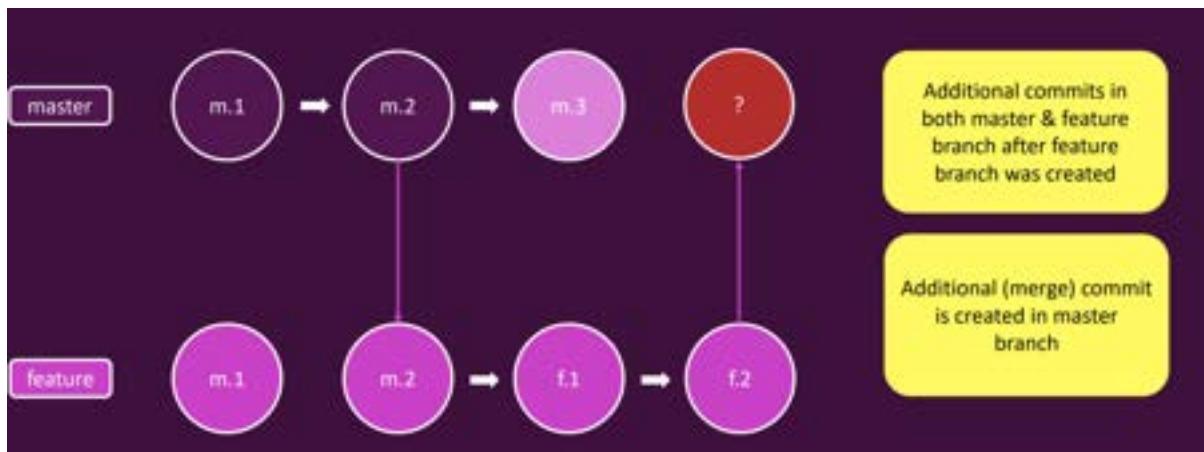


Fig. Shows Master & Feature – Recursive Merge("Non -Fast-Forward")

```
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 3bc4238bd61db54eb276b6a474ac63c232fe0909 (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 76e4b63e5cb83abb78c0f5e1b61f2e1c11b4754a (feature)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:51:35 2022 +0530

    f2 added

commit f07ff2718002060f6icd541f9c635b5ab6edbb26
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:58:58 2022 +0530

    f1 added

commit f32b9469d954ae7e4b2082bf5f14435494697f390
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:17:14 2022 +0530

    m2 added

commit 09ac428c3a0b18858aec1c2f611898d8e42ab706
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 76e4b63e5cb83a0bb78c0f5e1861f2e1c11b4754a (feature)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:51:35 2022 +0530

    f2 added

commit fb7ff2710002060f6dc541f9c635b5ab6ed6b26
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:50:58 2022 +0530

    f1 added

commit f32b9469d954ae7e4b2062b9f14435494697f390
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:17:14 2022 +0530

    m2 added

commit 89ac428c3a0b18850aec1c2f611898d0e42ab706
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 f32b9469d954ae7e4b2062b9f14435494697f390 (HEAD -> master)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:17:14 2022 +0530

    m2 added

commit 89ac428c3a0b18850aec1c2f611898d0e42ab706
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date:   Fri May 20 23:16:24 2022 +0530

    m1 added
```

Fig. Error resolved

The screenshot shows the VS Code interface with the 'feature' branch selected in the 'BRANCHES' sidebar. The terminal tab is active, displaying the following commit history:

```
Author: Saif Panjesta <98874394+SaifPanjesta@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 76e4b63e5cb43abb78c0f5e1861f2e1c11b4754a (HEAD -> feature)
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date: Fri May 20 23:51:35 2022 +0530

f2 added

commit fb7ff2718002060f6dc0541f9c635b5ab6edbb26
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date: Fri May 20 23:50:58 2022 +0530

f1 added
```

Fig. feature switched commit history is same in current branch

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

The screenshot shows the VS Code interface with the 'master' branch selected in the 'BRANCHES' sidebar. The terminal tab is active, displaying the following output:

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

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

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

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

Fig. Successful created m3.txt file

The screenshot shows the VS Code interface with the 'feature' branch selected in the 'BRANCHES' sidebar. The terminal tab is active, displaying the following output:

```
D:\one\drive\data\Desktop\Branches>git merge feature
Merge made by the 'ort' strategy.
  Feature/f1.txt | 0
  Feature/F2.txt | 0
  2 files changed, 0 insertions(+), 0 deletions(-)
  create mode 100644 feature/f1.txt
  create mode 100644 feature/F2.txt

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

Fig. Successful merge feature branch into master

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

```
D:\One Drive\data\Desktop\Branches>git log
```

Commit details for the master branch:

- commit 7c39bb6e630d4f972c76ed54b8e4d47311e9eeef (HEAD -> master)
Merge: 07e9d09 76e4b63
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date: Sat May 21 01:09:28 2022 +0530
- Merge branch 'feature'
- commit 07e9d09bc9a7726064ec9d2cc0685058c45d78ce
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date: Sat May 21 01:07:36 2022 +0530
- m3 added
- commit 76e4b63e5cb83abb7c0f5e1861f2e1c11b4754a (feature)
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date: Fri May 20 23:51:35 2022 +0530
- f2 added
- commit #b7ff2718002060f6dc541f9c635b5ab6edb26
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date: Fri May 20 23:50:58 2022 +0530
- f1 added
- commit f32e9469d954ae7e4b2082b9f14435494697f390
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date: Fri May 20 23:17:14 2022 +0530
- m2 added

Left sidebar navigation includes: EXPLORER, OPEN EDITORS, BRANCHES (with branches m1.txt, m2.txt, m3.txt), OUTLINE, and TIMELINE.

Fig. full commit history shows in master branch

Let's do again –squash

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

```
D:\One Drive\data\Desktop\Branches>git reset --hard HEAD~1
```

HEAD is now at 07e9d09 m3 added

```
D:\One Drive\data\Desktop\Branches>git log
```

Commit details for the master branch:

- commit 07e9d09bc9a7726064ec9d2cc0685058c45d78ce (HEAD -> master)
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date: Sat May 21 01:07:36 2022 +0530
- m3 added
- commit f32e9469d954ae7e4b2082b9f14435494697f390
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date: Fri May 20 23:17:14 2022 +0530
- m2 added
- commit 09ac42bc3a8d18850aec1c2f6118980e42ab706
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date: Fri May 20 23:16:24 2022 +0530
- m1 added

Left sidebar navigation includes: EXPLORER, OPEN EDITORS, BRANCHES (with branches m1.txt, m2.txt, m3.txt), OUTLINE, and TIMELINE.

Fig. switch to previous commit

The screenshot shows the VS Code interface with the terminal tab selected. The terminal window displays the following git log 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 07e1d0bcfa7726808ec9d2cc8685058c45d78ce (HEAD -> master)
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date:   Sat May 21 01:07:36 2022 +0530

    m3 added.

Commit f32b0469d9543e7e4b2082b9f14435404607f198
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date:   Fri May 20 23:17:14 2022 +0530

    m2 added.

commit 09ac428c3a0b18858aec1c2f611898d8e42ab706
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date:   Fri May 20 23:16:24 2022 +0530

    m1 added.

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

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

The screenshot shows the VS Code interface with the terminal tab selected. The terminal window displays the following git commit and status output:

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

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

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

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

Fig. Committed Master and Feature Merge

```

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

    Master and Feature Merged

commit e7edde9bc9a7726060ec9d2cc0685a58c45d78ce
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 @9ac428c3a0b18850aec1c2f611898d0e42ab706
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

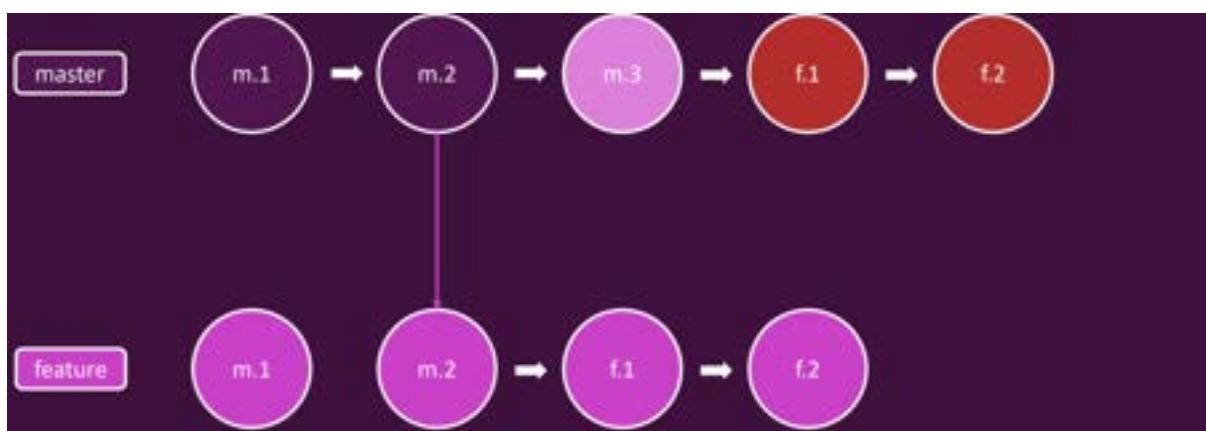


Fig. Shows Master & Feature of rebase

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.

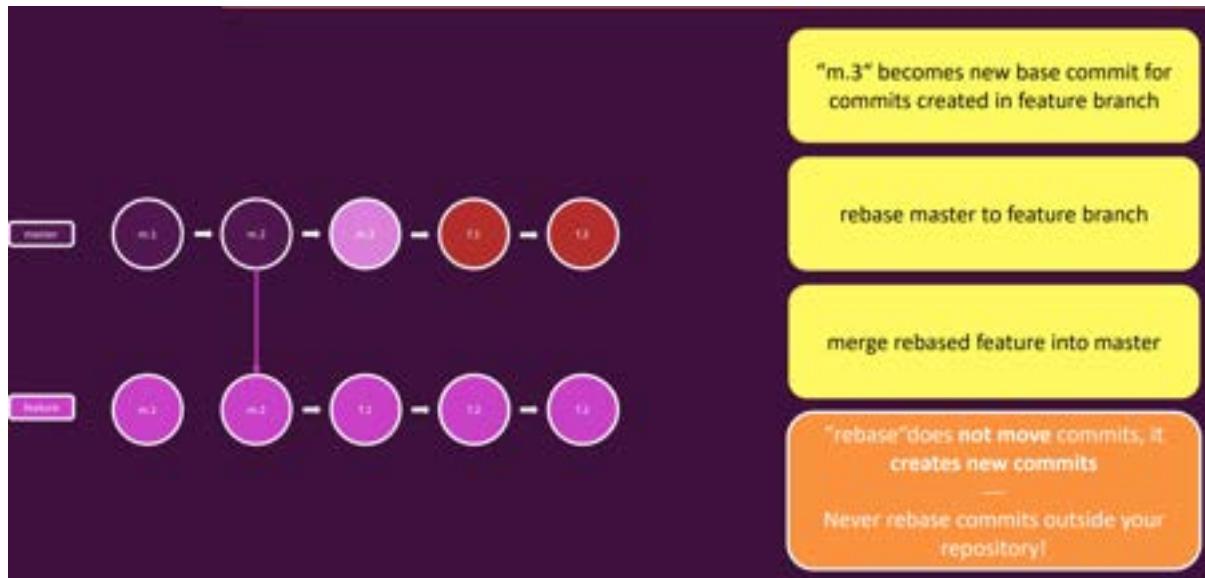


Fig. Shows using of rebase

Let's start rebasing

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

- EXPLORER:** Shows files `m1.txt`, `m2.txt`, and `m3.txt` under the `master` branch.
- BRAHES:** Shows the `master` branch with files `m1.txt`, `m2.txt`, and `m3.txt`.
- TERMINAL:** Displays the output of `git log` for the `master` branch, showing the following commits:

```
D:\One Drive\DATA\Desktop\Branches>git log
commit 797897e5dd3039dc14fbalec0166799e23a9275 (HEAD -> master)
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date:   Sat May 21 17:55:04 2022 +0530

    m3 added

commit 4fb0560cc444aeb58c835a2b2cf7eb955bc32651
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date:   Sat May 21 17:42:55 2022 +0530

    m2 added

commit e334ac365705adc4b4cce713e2307df8438eabcd
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date:   Sat May 21 17:41:45 2022 +0530

    m1 added
```

The terminal also shows the current working directory as `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 782e698a12faed175351a19je72bbfe1bd595de (HEAD -> feature)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date: Sat May 21 17:46:38 2022 +0530

    f2 added

commit 36fc7615d0b1740b5eb2ccfa3c3f428ae92eeb8
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date: Sat May 21 17:44:13 2022 +0530

    f1 added

commit 4fb056bc444aebe56c835a2b2cf7eb955bc32651
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date: Sat May 21 17:42:55 2022 +0530

    m2 added

commit e334ac365705a8c4b4cce713e2387dfb438eabe8
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 5e29effc53ffe00desd67f7f1565e296b415300 (HEAD -> feature)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date: Sat May 21 17:46:38 2022 +0530

    f2 added

commit 8a9a66cb85bd9139a4dd335d227124887c35fb1c3
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date: Sat May 21 17:44:13 2022 +0530

    f1 added

commit 782e698a12faed175351a19je72bbfe1bd595de (master)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date: Sat May 21 17:55:04 2022 +0530

    m3 added

commit 4fb056bc444aebe56c835a2b2cf7eb955bc32651
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date: Sat May 21 17:42:55 2022 +0530

    m2 added

commit e334ac365705a8c4b4cce713e2387dfb438eabe8
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

A screenshot of a terminal window titled "TERMINAL". The command entered is "git merge feature". The output shows:

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

Fig. fast-forward merging

When to apply rebase?

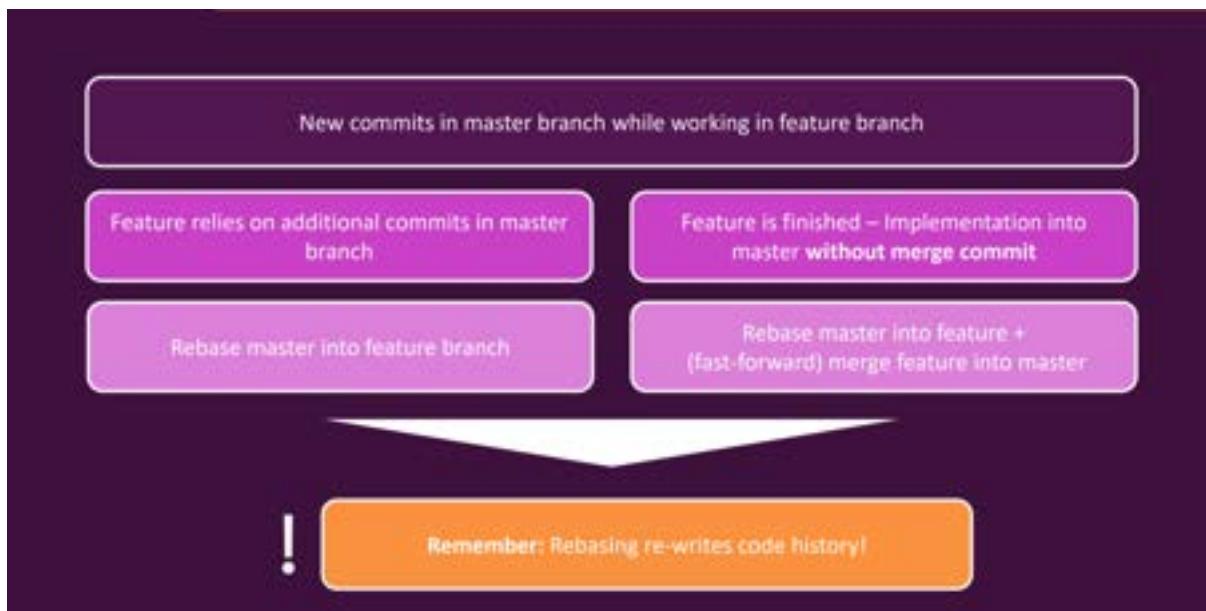


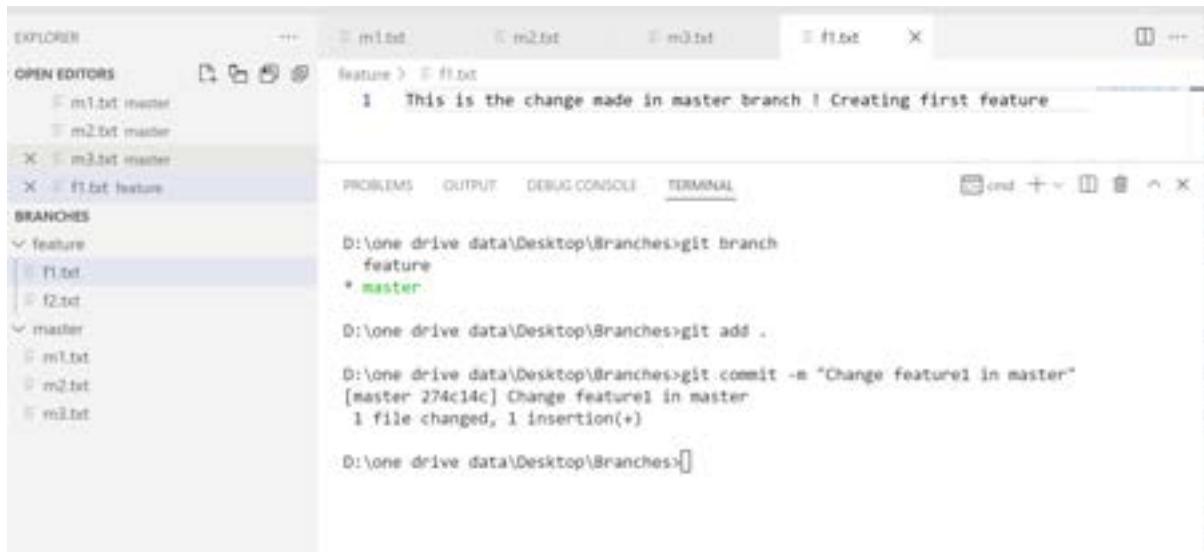
Fig. Workflow

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

Handling Merge Conflicts:

How to fix conflicts during the merge?

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



The screenshot shows the VS Code interface with the 'feature' branch selected in the sidebar. The terminal shows the command 'git commit -m "Change feature1 in master"' being run, which adds a new file 'f1.txt' and commits it with the message 'Change feature1 in master'. The status bar at the bottom indicates '1 file changed, 1 insertion(+)'.

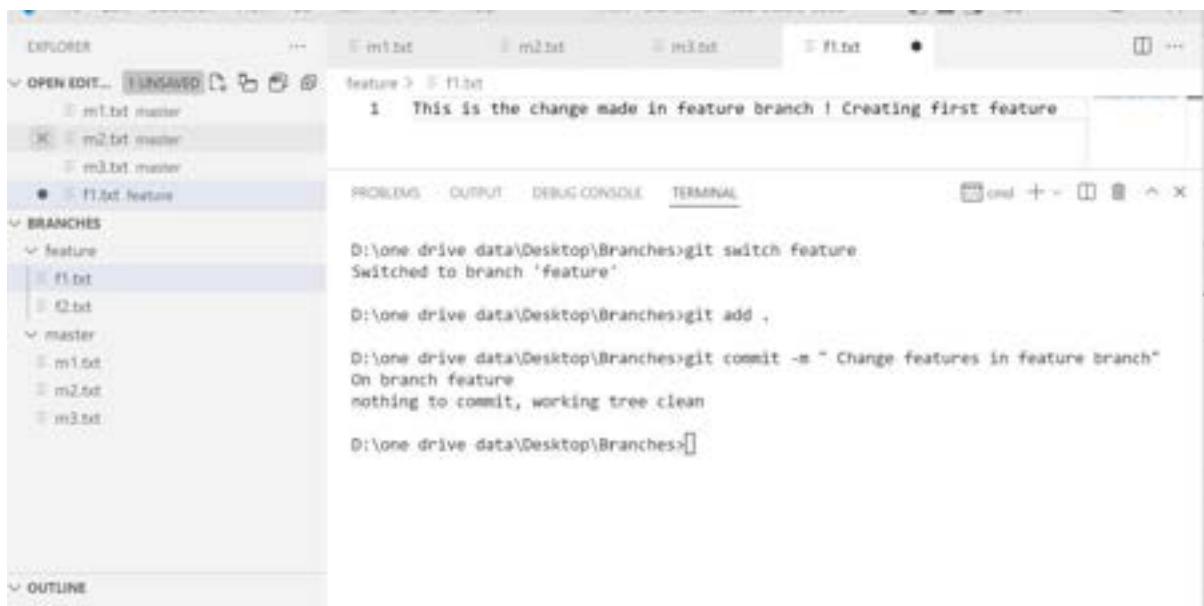
```
D:\One Drive Data\Desktop\Branches>git branch
  feature
* master

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

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

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

Fig. Shows Working feature in Master Branch



The screenshot shows the VS Code interface with the 'feature' branch selected in the sidebar. The terminal shows the command 'git switch feature' being run, which switches the working directory to the 'feature' branch. The status bar at the bottom indicates 'Switched to branch 'feature''. The terminal also shows the command 'git add .' and 'git commit -m " Change features in feature branch" On branch feature nothing to commit, working tree clean'.

```
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 branches m1.txt, m2.txt, m3.txt, and f1.txt feature.
- OPEN EDITORS**: Shows f1.txt feature with content:


```

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


```

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

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

D:\one drive data\Desktop\Branches>[]
      
```
- OUTLINE**: Shows no symbols found in document f1.txt.
- TIMELINE**: Shows a timeline of changes.

Fig. Shows Conflicts Created

How to handle these conflicts?

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

- EXPLORER**: Shows branches m1.txt, m2.txt, m3.txt, and f1.txt feature.
- OPEN EDITORS**: Shows f1.txt feature with content identical to the previous screenshot.
- BRANCHES**: Shows feature branch with f1.txt selected.
- TERMINAL**: Shows command output:


```

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

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

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

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

Fig. git status shows how to handle conflicts

The screenshot shows 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 the following content:


```
1 | Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
1 | <<<<< HEAD (Current Change)
2 | This is the change made in master branch ! Creating first feature
3 | =====
4 | This is the change made in feature branch ! Creating first feature
5 | >>>> feature (Incoming Change)
```
- BRANCHES**: Shows branches master, m1.txt, m2.txt, m3.txt, and feature (f1.txt selected).
- TERMINAL**: Shows the command output:


```
D:\One Drive Data\Desktop\Branches>git merge feature
Auto-merging f1.txt
CONFLICT (content): Merge conflict in feature/f1.txt
Automatic merge failed; fix conflicts and then commit the result.
```
- PROBLEMS**, **OUTPUT**, **DEBUG CONSOLE**: Standard VS Code status bars.

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 the following content:


```
1 | Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
1 | <<<<< HEAD (Current Change)
2 | This is the change made in master branch ! Creating first feature
3 | =====
4 | This is the change made in feature branch ! Creating first feature
5 | >>>> feature (Incoming Change)
```
- BRANCHES**: Shows branches master, m1.txt, m2.txt, m3.txt, and feature (f1.txt selected).
- TERMINAL**: Shows the command output:


```
D:\One Drive Data\Desktop\Branches>git diff
diff --cc feature/f1.txt
index b4bBadd,B2fd67..0000000
--- a/feature/f1.txt
+++ b/feature/f1.txt
@@@ -1,1 +1,5 @@
- This is the change made in master branch ! Creating first feature
+ This is the change made in feature branch ! Creating first feature
+<<<<< HEAD
++This is the change made in master branch ! Creating first feature
+=====+
++This is the change made in feature branch ! Creating first feature
+>>>> feature
```
- PROBLEMS**, **OUTPUT**, **DEBUG CONSOLE**: Standard VS Code status bars.

Fig. git diff show how to handle conflicts

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

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

- EXPLORER:** Shows files m1.txt, m2.txt, m3.txt under master, and f1.txt under feature.
- OPEN EDITORS:** Shows f1.txt feature with the content "This is the change made in master branch ! Creating first feature".
- BRANCHES:** Shows branches feature (with f1.txt and f2.txt) and master (with m1.txt, m2.txt, m3.txt).
- OUTLINE:** Shows no symbols found in document f1.txt.
- TERMINAL:** Displays the following command-line session:

```
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>
```

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, and f1.txt.
- OPEN EDITORS**: Shows f1.txt with the status "UNSAVED". The content of f1.txt is:


```
1 This is the change made in master branch | Creating first feature
2
```
- BRANCHES**: Shows branches feature (containing f1.txt and f2.txt) and master (containing m1.txt, m2.txt, m3.txt).
- PROBLEMS**: Displays a git status message:


```
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)
```
- OUTPUT**: Shows terminal output of git commands:


```
D:\One Drive Data\Desktop\Branches>git add .
D:\One Drive Data\Desktop\Branches>git commit -m "merged feature and master in f1 file"
[master 1755d3f] merged feature and master in f1 file
D:\One Drive Data\Desktop\Branches>[ ]
```
- TIMELINE**: Shows the history of changes made to f1.txt.

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, and f1.txt.
- OPEN EDITORS**: Shows f1.txt with the status "UNSAVED". The content of f1.txt is:


```
merged feature and master in f1 file
```
- BRANCHES**: Shows branches feature (containing f1.txt and f2.txt) and master (containing m1.txt, m2.txt, m3.txt).
- PROBLEMS**: Displays a git log message:


```
D:\One Drive Data\Desktop\Branches>git log
commit 1755d3f086d5384f3d4a3a1309a7ae4d498946cc (HEAD -> master)
Merge: 274c14c 489837b
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date:   Sat May 21 19:05:13 2022 +0530
```
- OUTPUT**: Shows terminal output of git commands:


```
Changes features in feature branch

commit 274c14c6c285597fdb16499fd942e0376b1e343f
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date:   Sat May 21 18:46:10 2022 +0530
```
- TIMELINE**: Shows the history of changes made to f1.txt, including:
 - f1 added
 - f2 added
 - f1 added

Fig.History committed in master branch

Understanding “git cherry-pick”

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

Merge vs Rebase vs Cherry Pick



Fig. Quick recap

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

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

- EXPLORER:** Shows files `m1.txt`, `m2.txt`, `m3.txt`, and `ft.txt`. `m1.txt` is selected in the master branch.
- OPEN EDITORS:** Shows `m1.txt` with content: "Some Impotent thing in master".
- BRANCHES:** Shows branches `feature` and `master`. `m1.txt` is checked out under `master`.
- TERMINAL:** Shows the command 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

```

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 "f3 added"
[feature 26513b2] f3 added
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 feature/f3.txt

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

```

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 Explorer, Problems, Output, Debug Console, and Terminal tabs. The Explorer sidebar shows branches: feature (f1.txt, f2.txt, f3.txt, f4.txt) and master (m1.txt, m2.txt, m3.txt). The terminal shows the git log for the feature branch:

```
D:\one drive data\Desktop\Branches>git log
commit b7f6257adf5941db359f0d5dd99890533a1c48d0 (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 47b617c854c8395deb775e5be7a828c7aa6577da
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 214c6f77ad89e731c5afffb1e70334c45a22c9ef
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 bc75b559f061ce06572a990834d36d2f7836a255 (master)
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date:   Sat May 21 19:32:52 2022 +0530
```

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 Explorer, Problems, Output, Debug Console, and Terminal tabs. The Explorer sidebar shows branches: feature (f1.txt, f2.txt, f3.txt, f4.txt) and master (m1.txt, m2.txt, m3.txt). The terminal shows the command to cherry-pick the specific commit from the feature branch into the master branch:

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

D:\one drive data\Desktop\Branches>git cherry-pick b7f6257adf5941db359f0d5dd99890533a1c48d0
[master 9620022] typo in m1 file added
Date: Sat May 21 19:47:01 2022 +0530
1 file changed, 1 insertion(+), 1 deletion(-)

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

Fig. specific commits has been added to master branch

```
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 b7f6257adf5941db359f0d5dd99890533a1c40d0 (HEAD -> feature)
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date: Sat May 21 19:47:01 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 374c14c6c285597fdb16499fd942e0370b1e343f
Author: Saif Panjesha <98874394+SaifPanjesha@users.noreply.github.com>
Date: Sat May 21 18:35:29 2022 +0530
```

Fig. git log shows that commit is successful.

Master branch:

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

typo in m1 file added

Feature branch:

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

typo in m1 file added

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

Working with “git tag”:

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

Lightweight tags:

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

```
D:\one drive data\Desktop\Branches>git tag  
D:\one drive data\Desktop\Branches>git tag 2.2 b7f6257adf5941db359f0d5d99890533a1c48  
@  
D:\one drive data\Desktop\Branches>git show 2.2  
commit b7f6257adf5941db359f0d5d99890533a1c4800 (HEAD -> feature, tag: 2.2)  
Author: Saif Panjesha <9887439445a1fPanjesha@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
```

Fig. Lightweight tags Example

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
cmd + ⌘ ⌘ ⌘ X

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 lists files and folders. It shows a tree structure with 'm1.txt master' at the root, followed by 'f3.txt feature', 'f4.txt feature', 'm2.txt master', 'm3.txt master', and 'f1.txt feature'. Under 'BRANCHES', there is a 'feature' branch containing 'f1.txt'. The right side of the interface is the Terminal pane, which displays the command-line history:

```
EXPLORER: ...
OPEN EDIT... UNGAVE... ⌂ ⌂ ⌂ ⌂ ⌂
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
cmd + ⌘ ⌘ ⌘ X

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

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

Fig. Successful tag deleted

Annotated tags:

stored as full objects in the Git database

The screenshot shows a terminal window with the following output:

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

Previous tags

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

Working on m1

diff --git a/master/m1.txt b/master/m1.txt
index e69de29..ad9c9e6 100644
--- a/master/m1.txt
+++ b/master/m1.txt
@@ -0,0 +1 @@
+Some important thing in master
\ No newline at end of file
D:\One Drive\DATA\Desktop\Branches>
```

Fig. Annotated tags

Wrap Up Git Diving Deeper:

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

Fig. Git Diving Deeper commands.

From local to remote understanding:



Fig. Git knowledge with GitHub in cloud

Module Overview:

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

What is GitHub?



Fig. shows about Git and GitHub.

How Git and GitHub are connected?

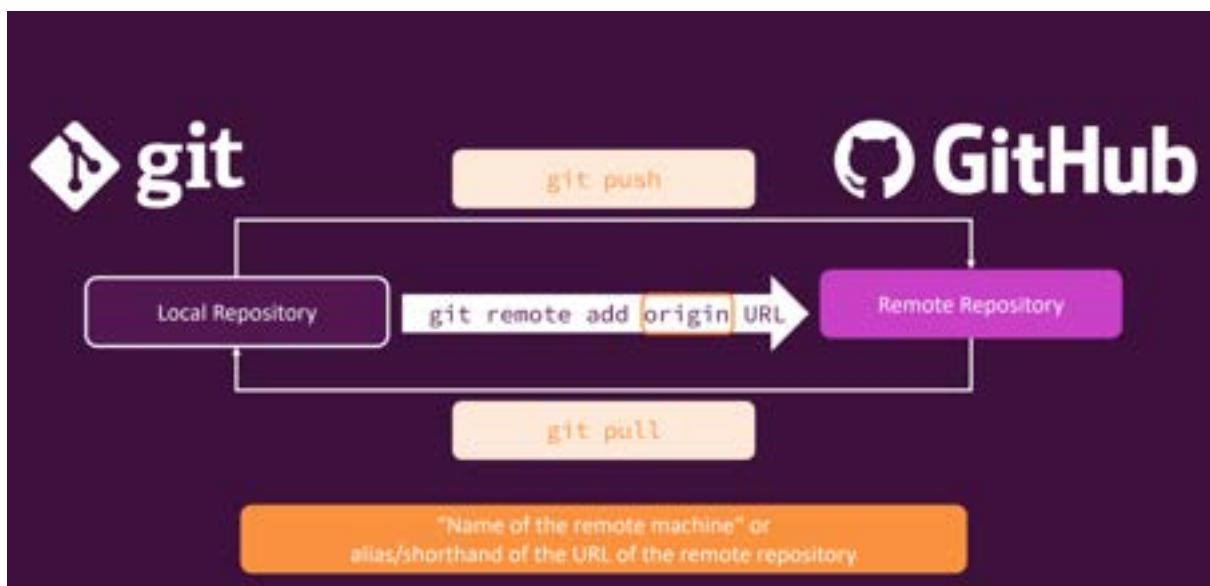


Fig. shows connecting git and GitHub - local to remote repository

Creating a GitHub account and introducing GitHub

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

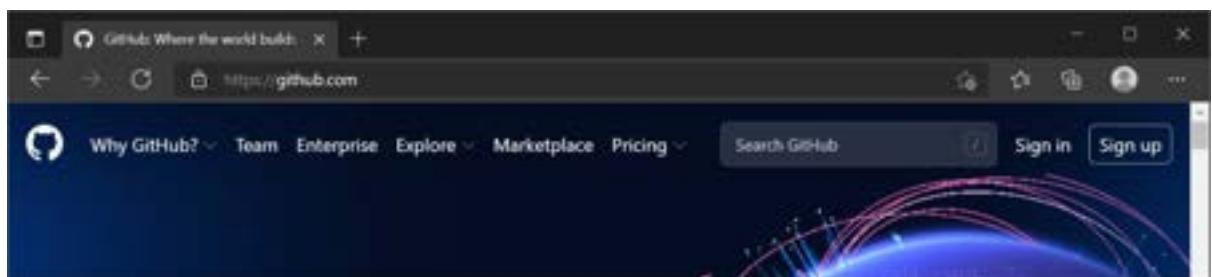


Fig. official site

2. Enter your email address.



Fig. Add your email address

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



Fig. Creation of Username & Password

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

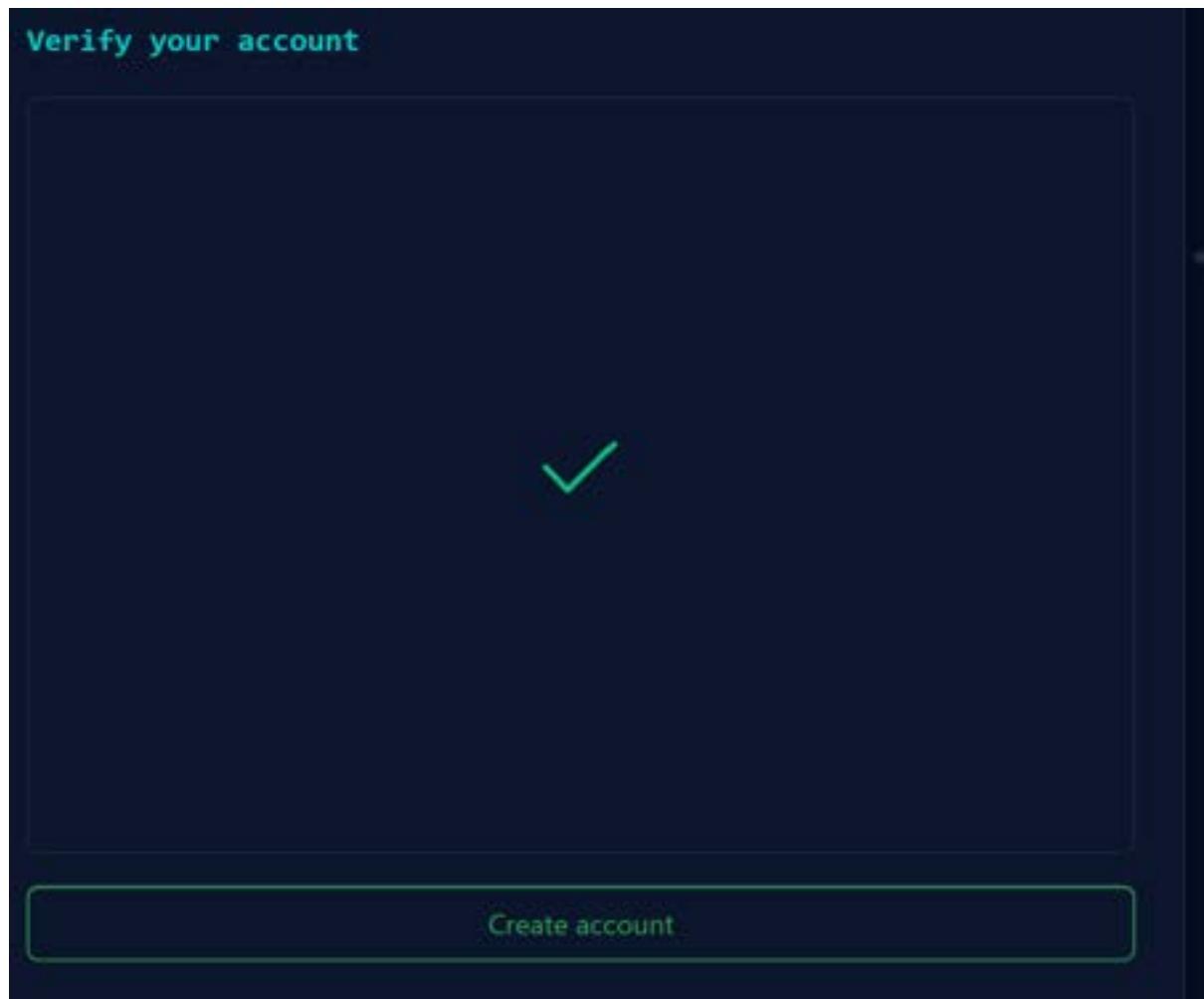


Fig. Account Verified

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

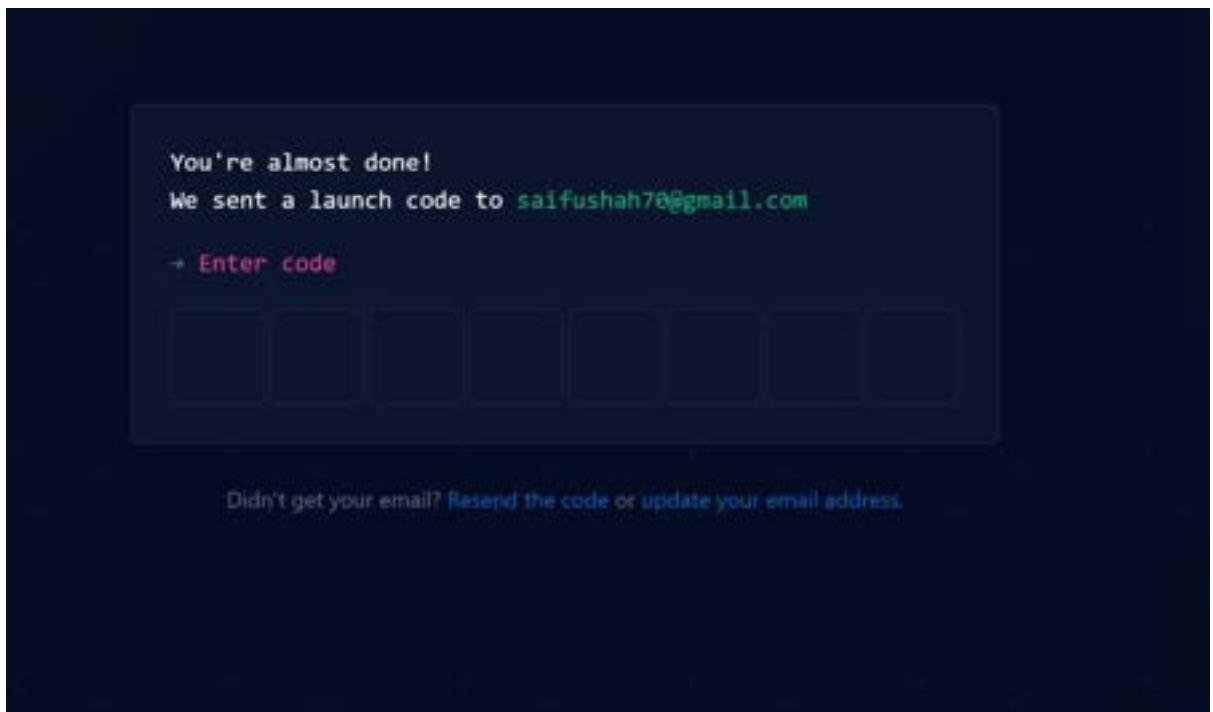


Fig. Enter the Code

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

Tip

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

GitHub opens a personalized page in your browser.

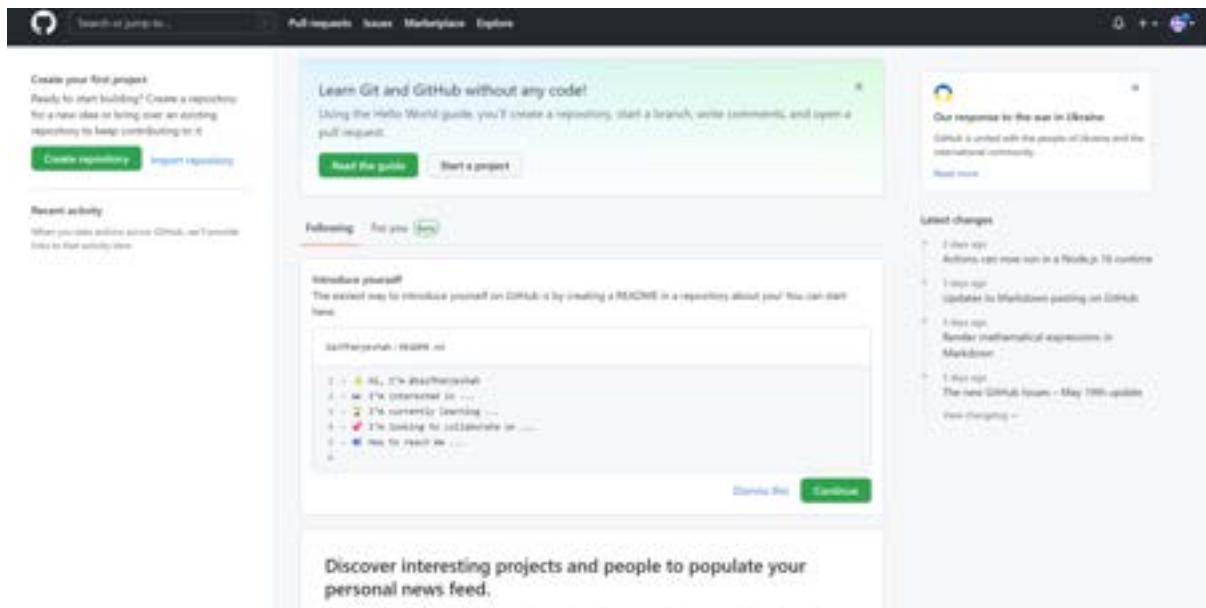


Fig. Account Creation Successful

Creating a repository:

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

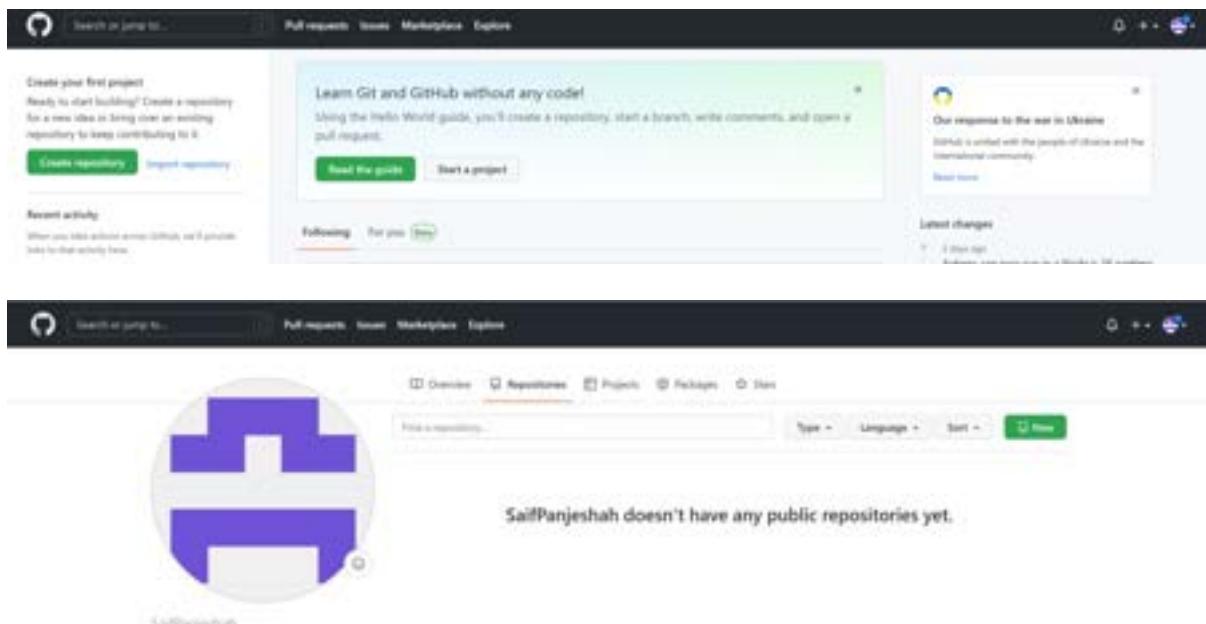


Fig. Creating repository

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 .

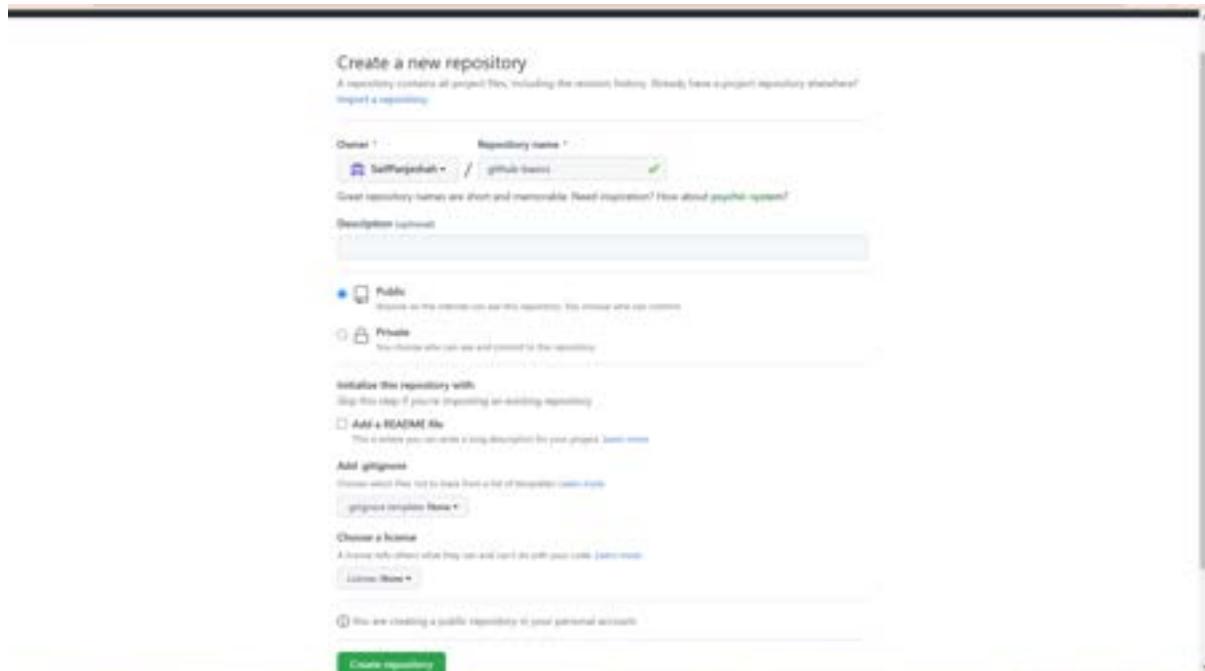


Fig. Created a new repository

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

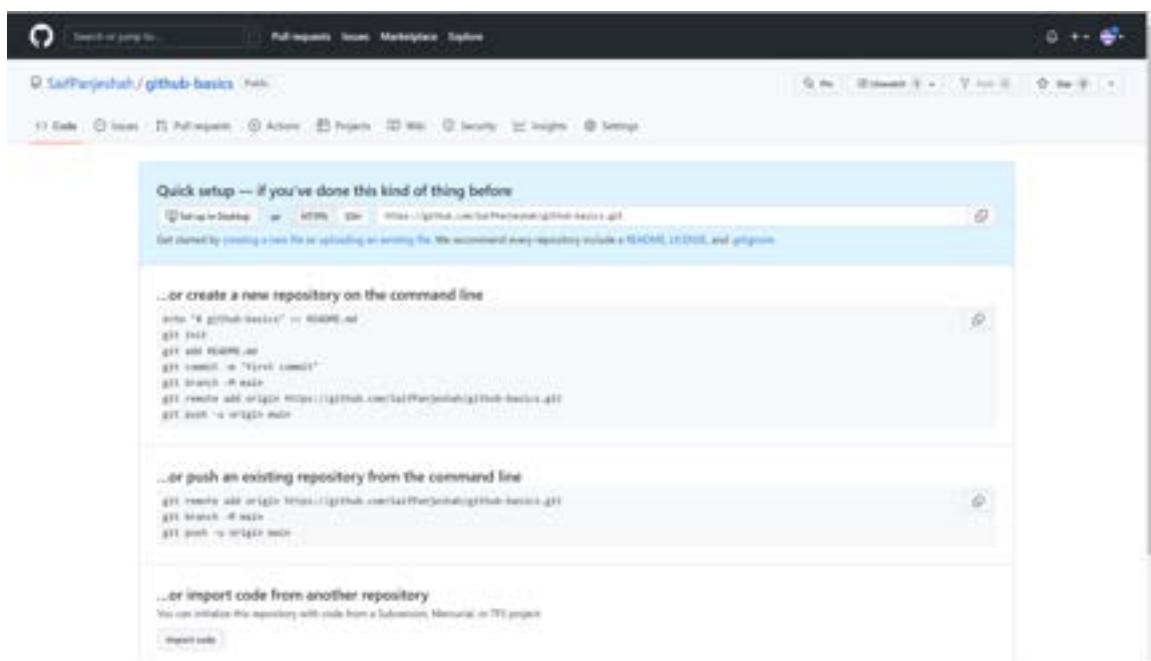


Fig. Setup git on GitHub

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

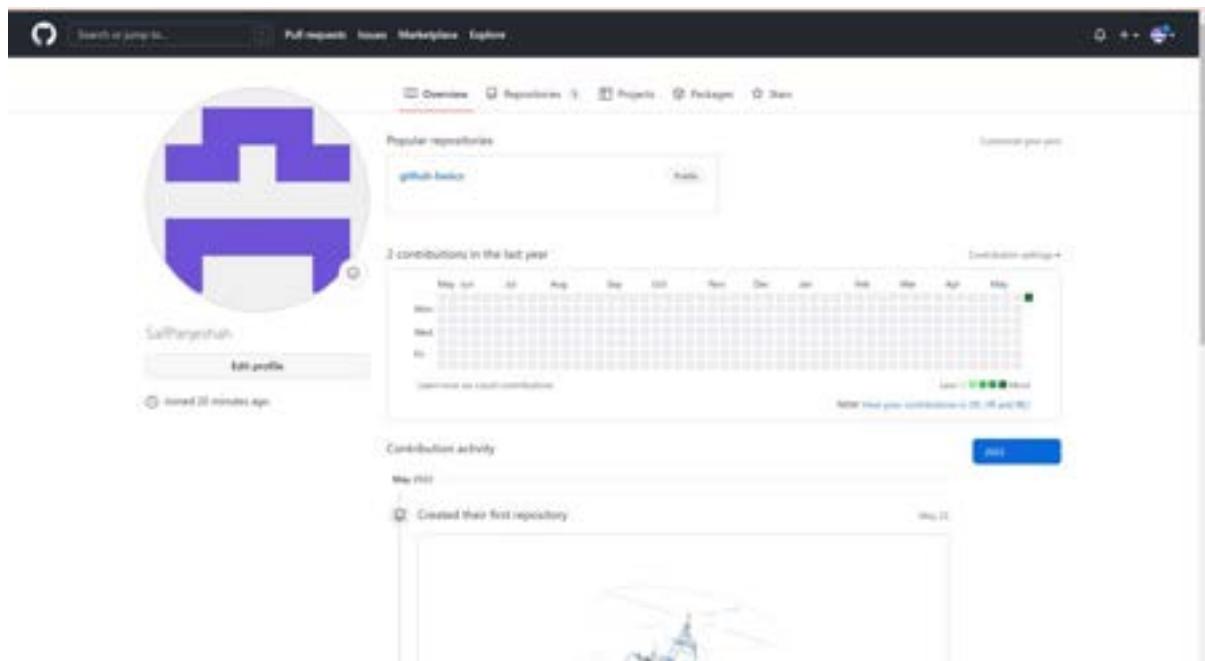


Fig. Successful remote repository added

Connecting Local and remote repository:

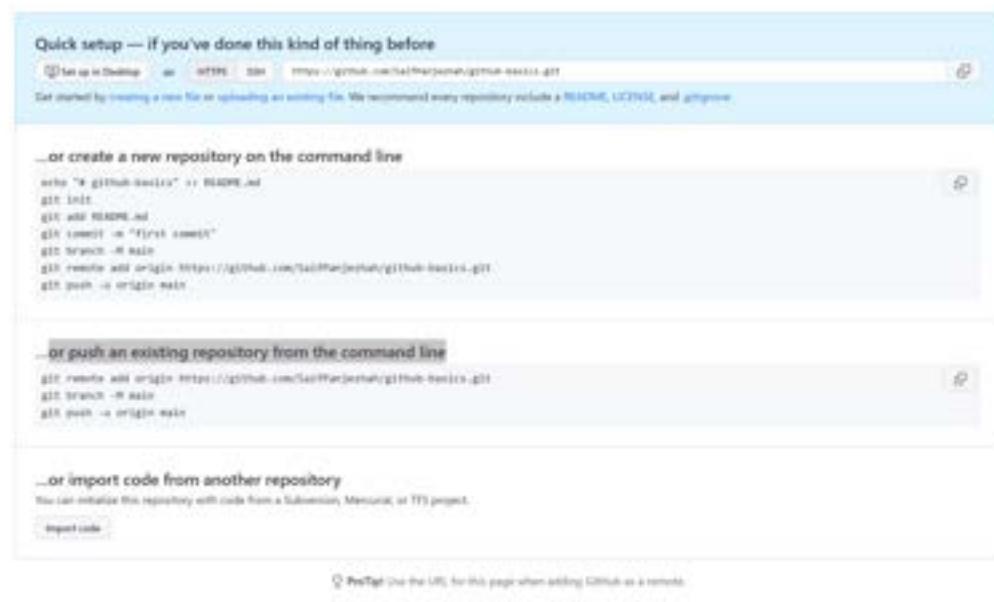


Fig. pushing repository from git

Git local repository

The screenshot shows the VS Code interface with the 'Terminal' tab active. The terminal output shows the following steps to create a local Git repository:

```
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

```
DIRECTORY: m1.txt X  
master > m1.txt  
1  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL  
D:\One Drive\DATA\Desktop\github-basics>git branch  
* master  
D:\One Drive\DATA\Desktop\github-basics>git log  
commit b9fe76d083da58314ad83a2fa13138d7dabf0294 (HEAD -> master)  
Author: Saif Panjeshah <98874394+SaifPanjeshah@users.noreply.github.com>  
Date: Sun May 22 17:58:15 2022 +0530  
m1 added  
D:\One Drive\DATA\Desktop\github-basics>
```

Fig. branches and commit history.

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

Using Command:

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

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

A screenshot of the Visual Studio Code interface. The Explorer sidebar shows a folder named 'GITHUB1-BASICS' containing 'master' and 'm1.txt'. A terminal tab is open, displaying 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

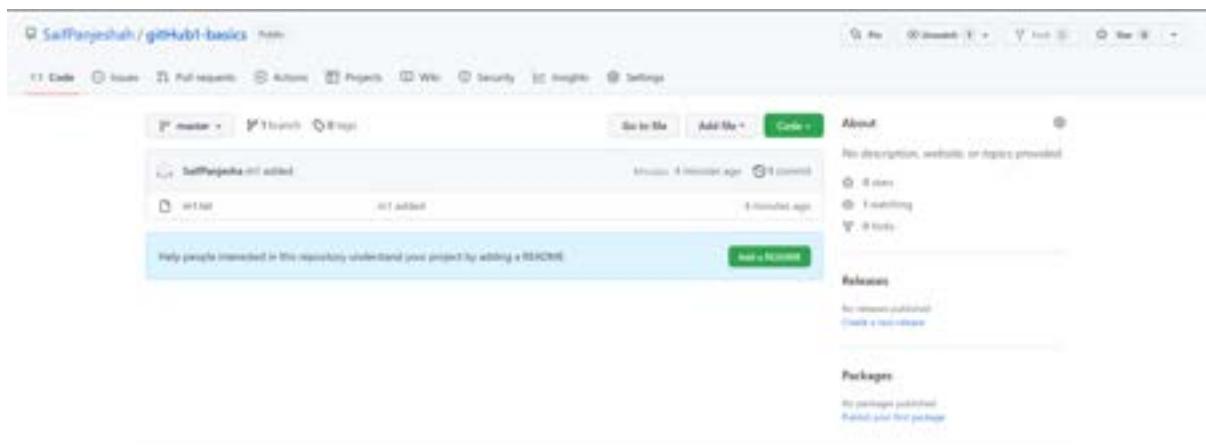


Fig. Successful push our local repository on GitHub via accessing through Internet browser.

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**.

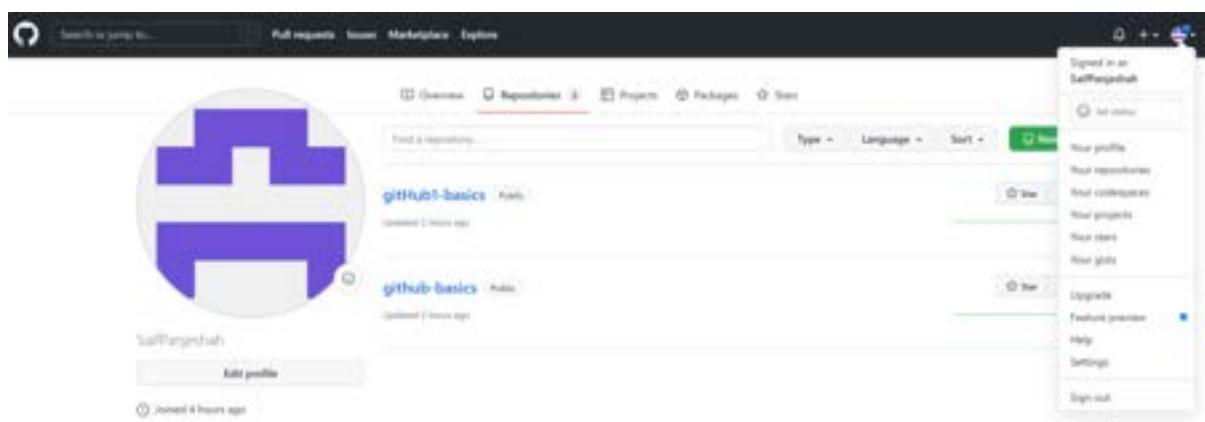


Fig. After verifying email address GitHub profile

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

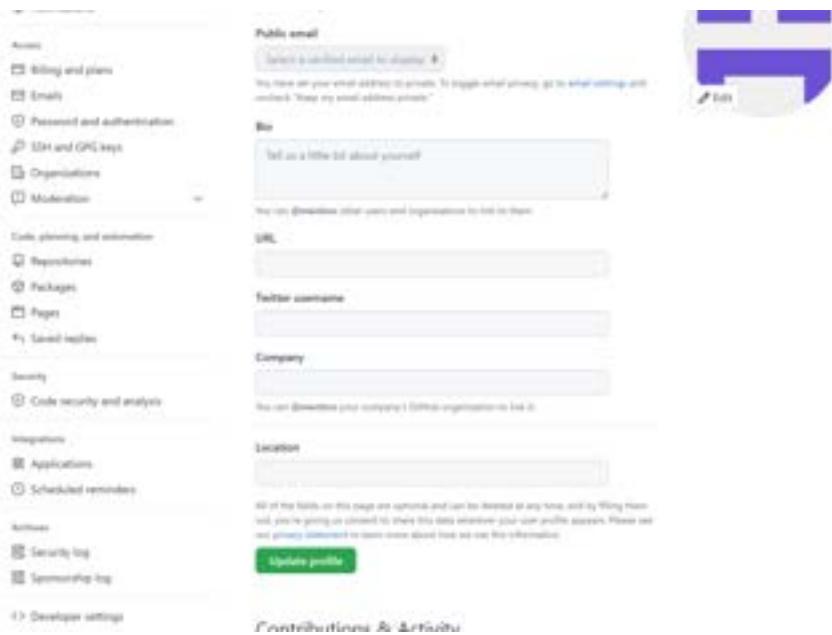


Fig. Setting to open developer setting

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

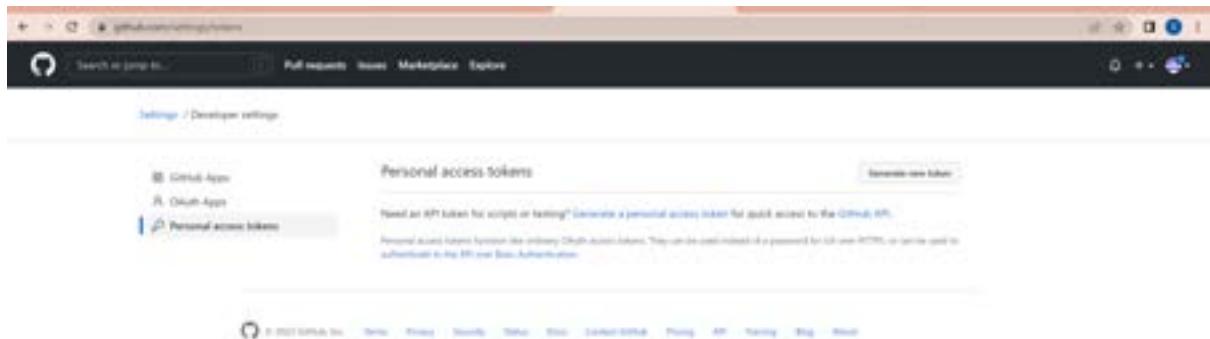


Fig. Personal access tokens

- e. Give your token a descriptive name.

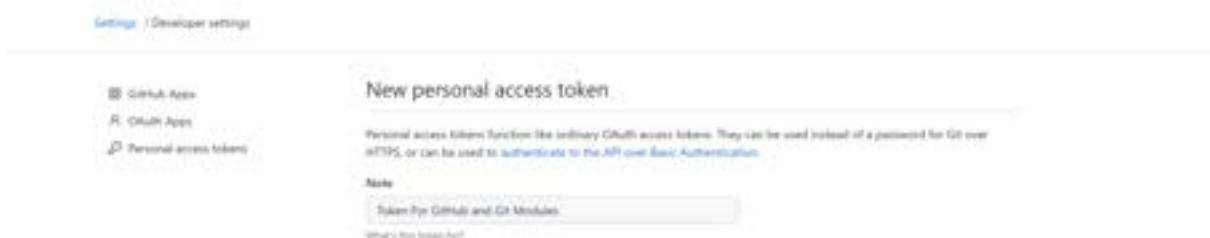


Fig. Descriptive Name of Token

- 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 type="checkbox"/> repo	Full control of private repositories
<input type="checkbox"/> repo:status	Access commit status
<input type="checkbox"/> repo_deployment	Access deployment status
<input type="checkbox"/> public_repo	Access public repositories
<input type="checkbox"/> repo:invite	Access repository invitations
<input type="checkbox"/> security_events	Read and write security events
<input checked="" type="checkbox"/> workflow	Update GitHub Action workflows
<input checked="" type="checkbox"/> write_packages	Upload packages to GitHub Package Registry
<input 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 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 Read ALL user profile data Access user email addresses (read-only) Follow and unfollow users
<input checked="" type="checkbox"/> delete_repo	Delete repositories
<input checked="" type="checkbox"/> write_discussion	Read and write team discussions Read team discussions
<input type="checkbox"/> admin:enterprise	Full control of enterprises Manage enterprise runners and runner-groups Read and write enterprise billing data Read enterprise profile data
<input type="checkbox"/> admin:gpg_key	Full control of public user GPG keys (Developer Preview) Write public user GPG keys Read public user GPG keys

Generate token **Cancel**

Fig. Generating Token

Personal access tokens

Tokens you have generated that can be used to access the GitHub API.

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

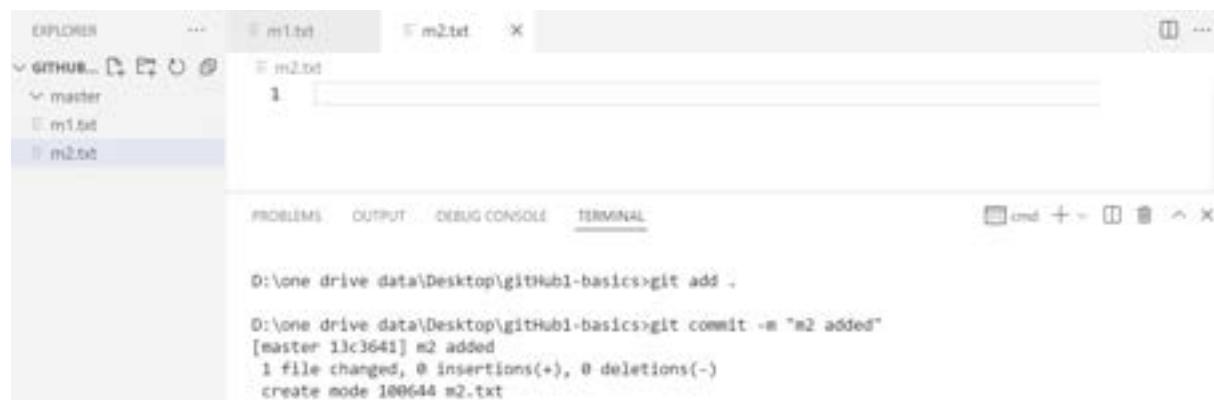
✓ d79c3e89vty043C94f711213abfd2118818878a79

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.

Delete

Fig. Token Successful Generated

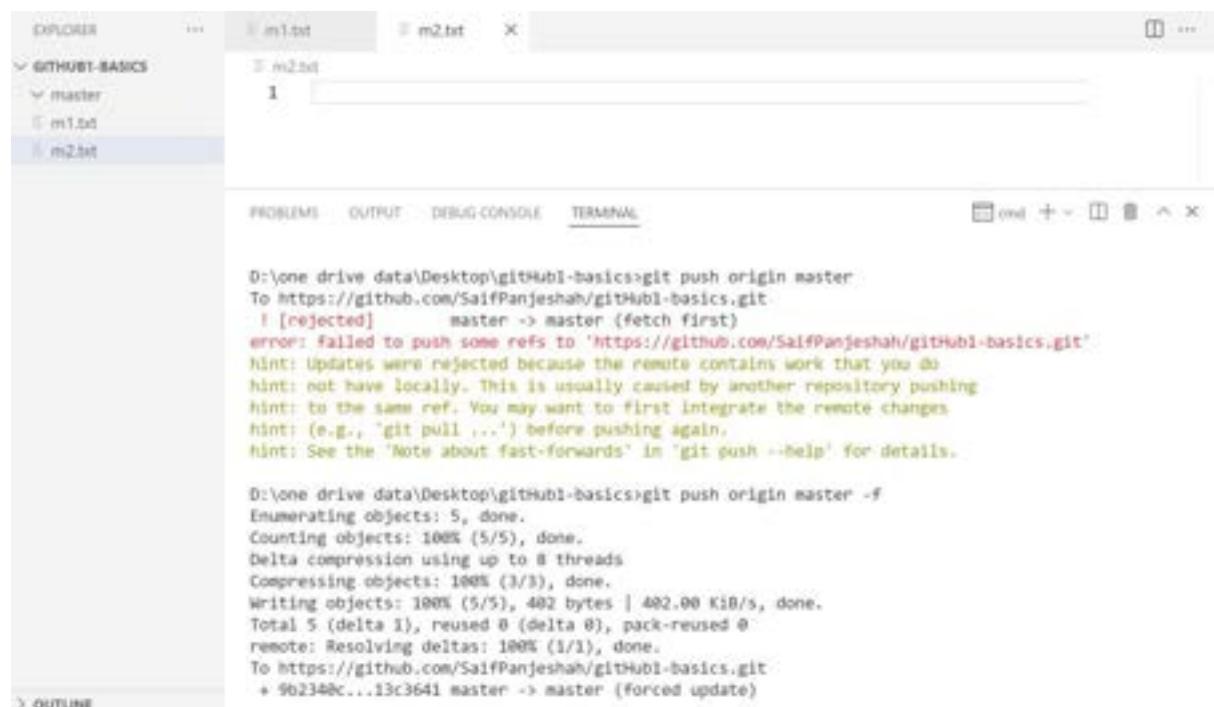
Pushing a Second Commit



A screenshot of the Visual Studio Code interface. The Explorer sidebar shows a repository named 'github1-basics' with a 'master' branch containing files 'm1.txt' and 'm2.txt'. The 'm2.txt' file is selected. The terminal tab shows the command line output:

```
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



A screenshot of the Visual Studio Code interface. The Explorer sidebar shows a repository named 'GitHub1-Basics' with a 'master' branch containing files 'm1.txt' and 'm2.txt'. The 'm2.txt' file is selected. The terminal tab shows the command line output:

```
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
 + 9b2348c...13c3641 master -> master (forced update)
```

Fig. Push M2 file to remote repository

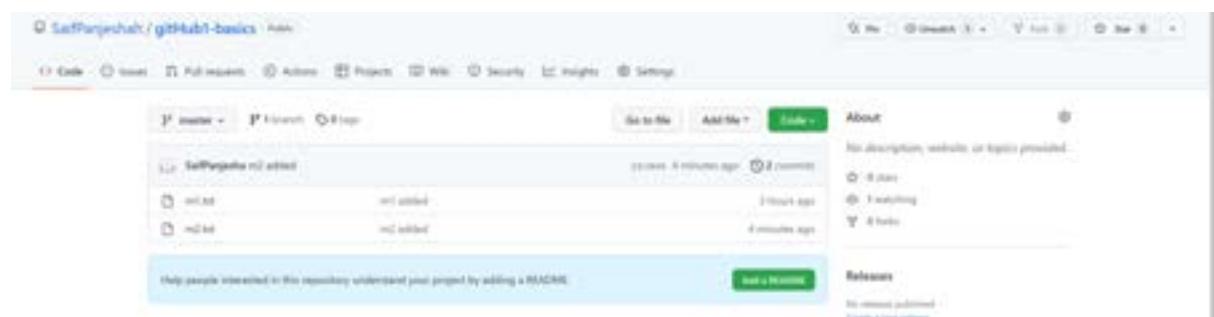
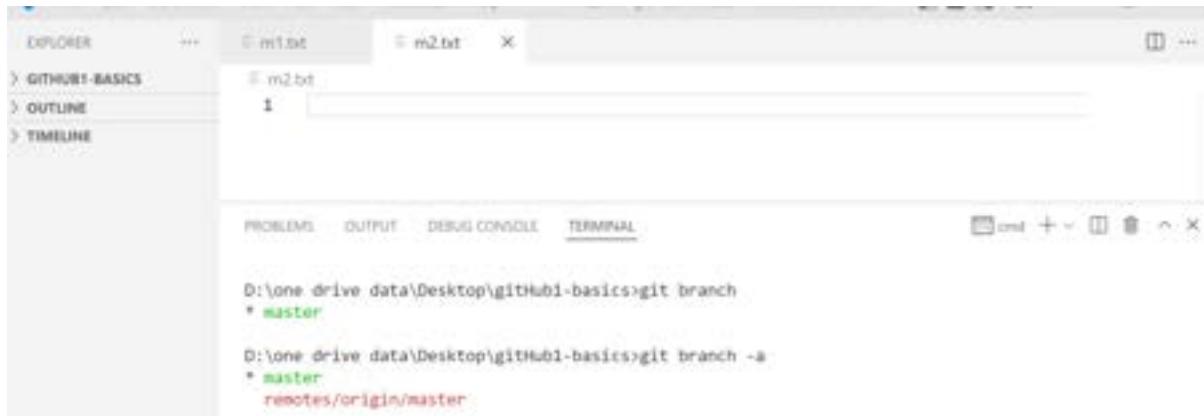


Fig. Successful on GitHub

From Local to remote Understanding the Workflow



A screenshot of the Visual Studio Code interface. The left sidebar shows 'EXPLORER', 'GITHUB-BASICS', 'OUTLINE', and 'TIMELINE'. The main area has tabs for 'm1.txt' and 'm2.txt'. Below the tabs is a code editor with the letter 'i'. At the bottom is a terminal window with the following text:

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

D:\One Drive\data\Desktop\github-basics>git branch -a
* master
  remotes/origin/master
```

Fig. added remote tracking branch

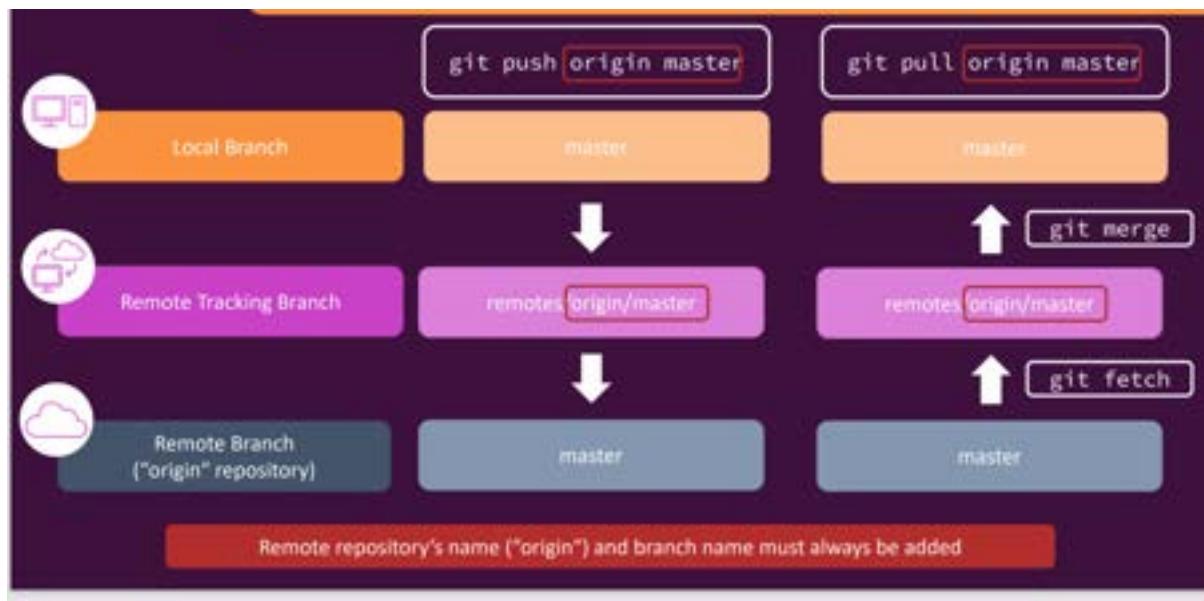
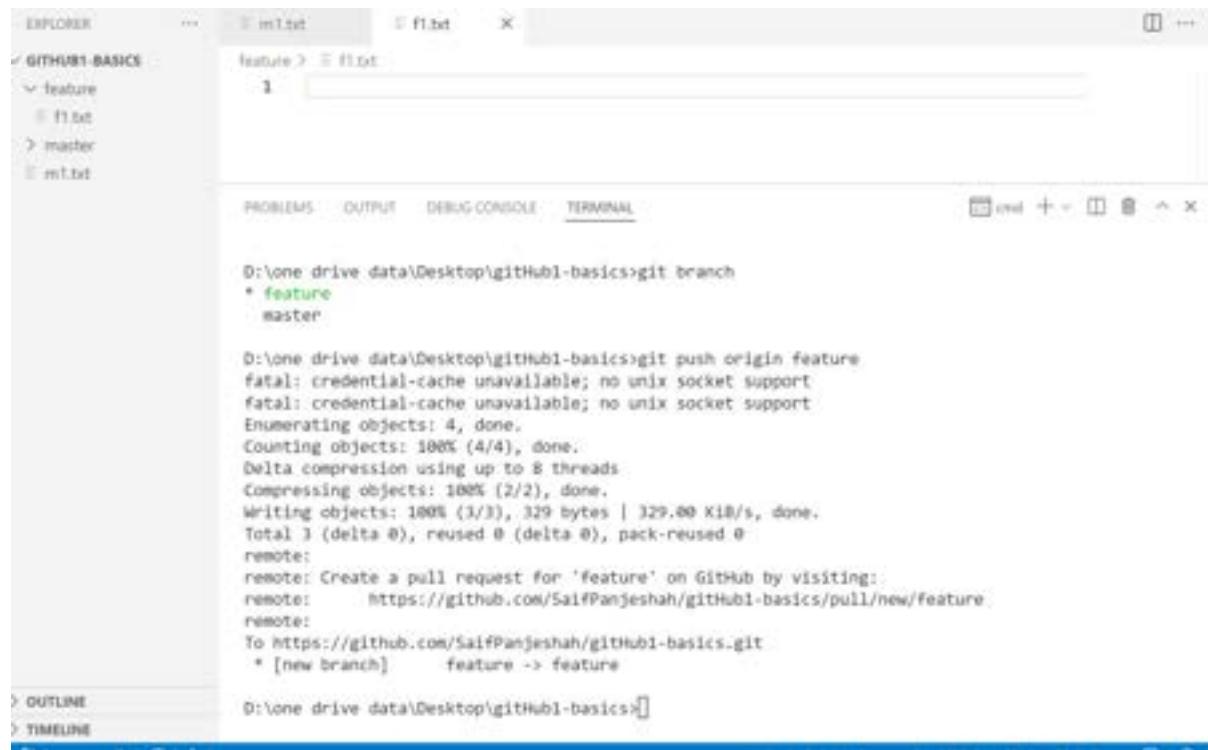


Fig. Local to remote Workflow

Remote Tracking Branches in Practices:



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

- EXPLORER**: Shows a folder named "GITHUB1-BASICS" containing branches "feature", "f1.txt", and "master".
- TERMINAL**: Displays the command 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
```
- OUTLINE** and **TIMELINE** panels are visible on the left.

Fig. Created New Feature Branch

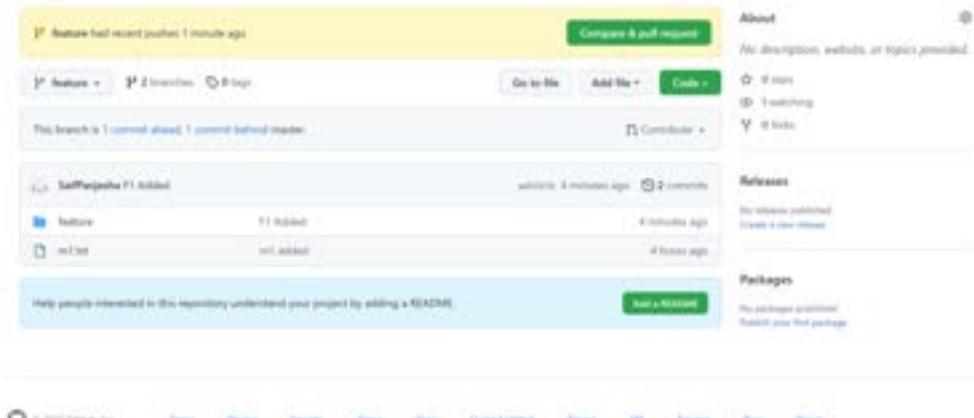


Fig. Successful added on GitHub

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

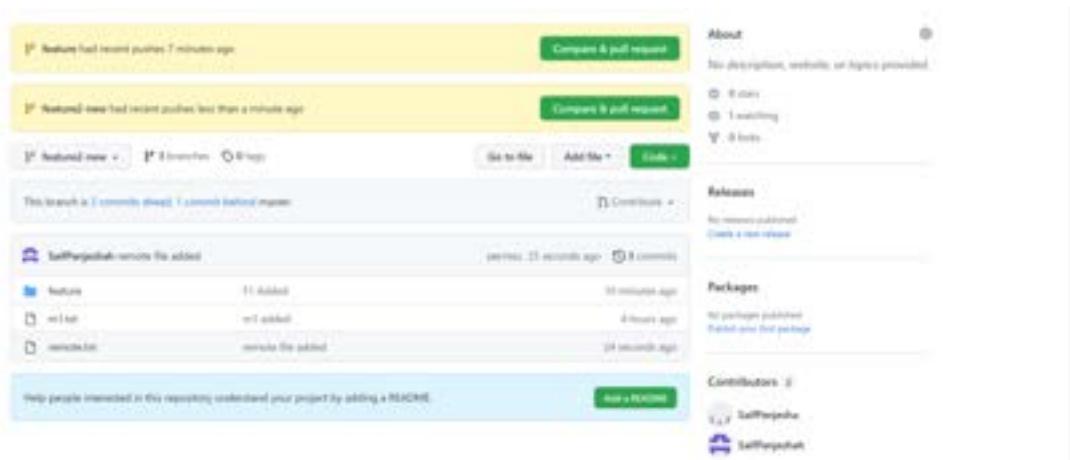


Fig. Created new Branch on Git Hub

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\GitHub-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\GitHub-Basics>git branch -a
* feature
  ls-remote
  master
  remotes/origin/feature
  remotes/origin/master

D:\One Drive\DATA\Desktop\GitHub-Basics>[ ]
```

Fig. Shows New Head Branch added in remote repository

How Can View this remote Branch **feature2-new** in git (local repository)?

Git fetch retrieves the latest state.

Git fetch origin works because it updates the remote tracking branch.

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

```
D:\One Drive\DATA\Desktop\GitHub-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\GitHub-Basics>git ls-remote
From https://github.com/SaifPanjeshah/github1-basics.git
13c36418a467824c7c0d92cf5883dc83277fb22c      HEAD
ad5257df399e536b96e5b6c056c81d6435c232c0      refs/heads/feature
68b796283e50d636a3de3e1d25859401ddc82a59      refs/heads/feature2-new
13c36418a467824c7c0d92cf5883dc83277fb22c      refs/heads/master
```

Fig. Shows git fetch remote retrieves latest state

The screenshot shows a terminal window with the following content:

```
EXPLORER PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
GITHUB-BASICS
> feature
* master
> 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

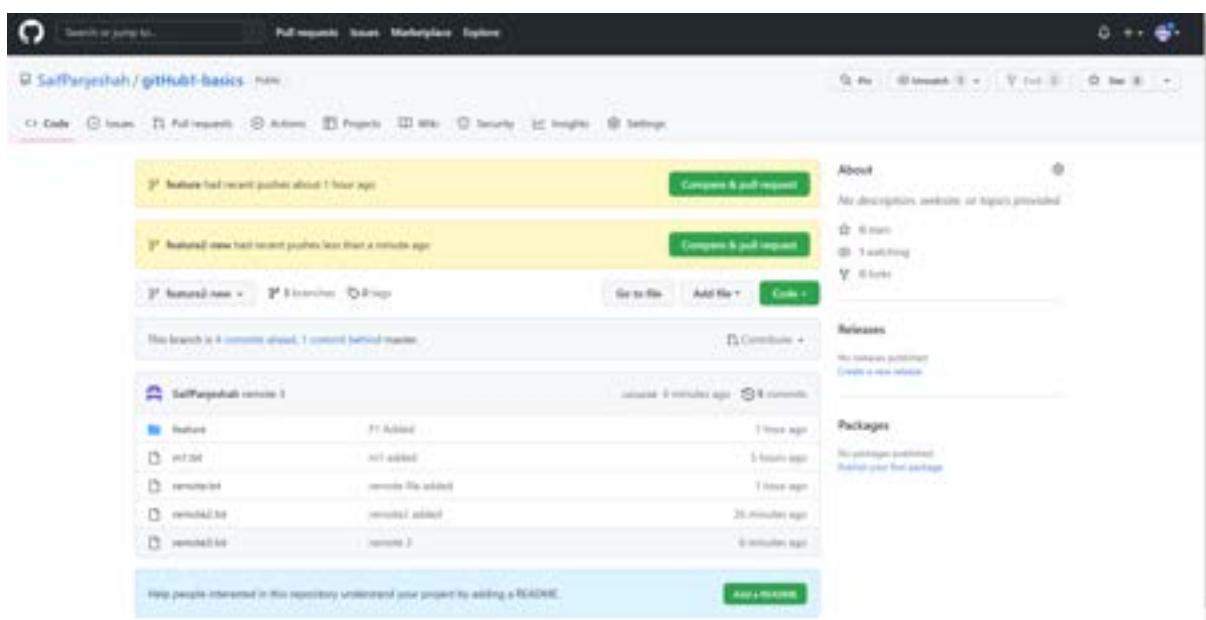


Fig. Shows Feature2 branch in GitHub

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)



D:\One Drive Data\Desktop\GitHub-Basics>git branch
feature
* master

D:\One Drive Data\Desktop\GitHub-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), 1.32 KB / 34.00 KB/s, done.
From https://github.com/SaifPanjeshah/GitHub-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\GitHub-Basics>git[]

Fig. Shows git pull successful added to local repository



D:\One Drive Data\Desktop\GitHub-Basics>git log
commit ca1a24814927cd7b19a9447218196931dff33e3f (HEAD, origin/feature2-new)
Author: SaifPanjeshah <106016819+SaifPanjeshah@users.noreply.github.com>
Date: Sun May 22 23:32:22 2022 +0530

Remote 3

commit 94916dc:a737b30e4401197affecf32f1b490ca78
Author: SaifPanjeshah <106016819+SaifPanjeshah@users.noreply.github.com>
Date: Sun May 22 23:32:18 2022 +0530

remote2 added

commit 68b796283e50d636a3de3e1d25859401ddc82a59
Author: SaifPanjeshah <106016819+SaifPanjeshah@users.noreply.github.com>
Date: Sun May 22 22:46:14 2022 +0530

remote file added

commit ad5257df399e536b96e5b6cd56c81d6435c232c0 (origin/feature)
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date: Sun May 22 22:13:16 2022 +0530

F1 Added

commit 97cc11cd8342a5c23a854b14db17f9e8ebed1c76
Author: Saif Panjesta <98874394+SaifPanjesta@users.noreply.github.com>
Date: Sun May 22 18:23:33 2022 +0530

m1 added

Fig. Shows full commit history of GitHub feature2 -new Branch in local repository

Understanding Local Tracking Branches:



Fig. Overview of Branches

A screenshot of a terminal window in a code editor interface. The terminal tab is active, showing the following command history:

```
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

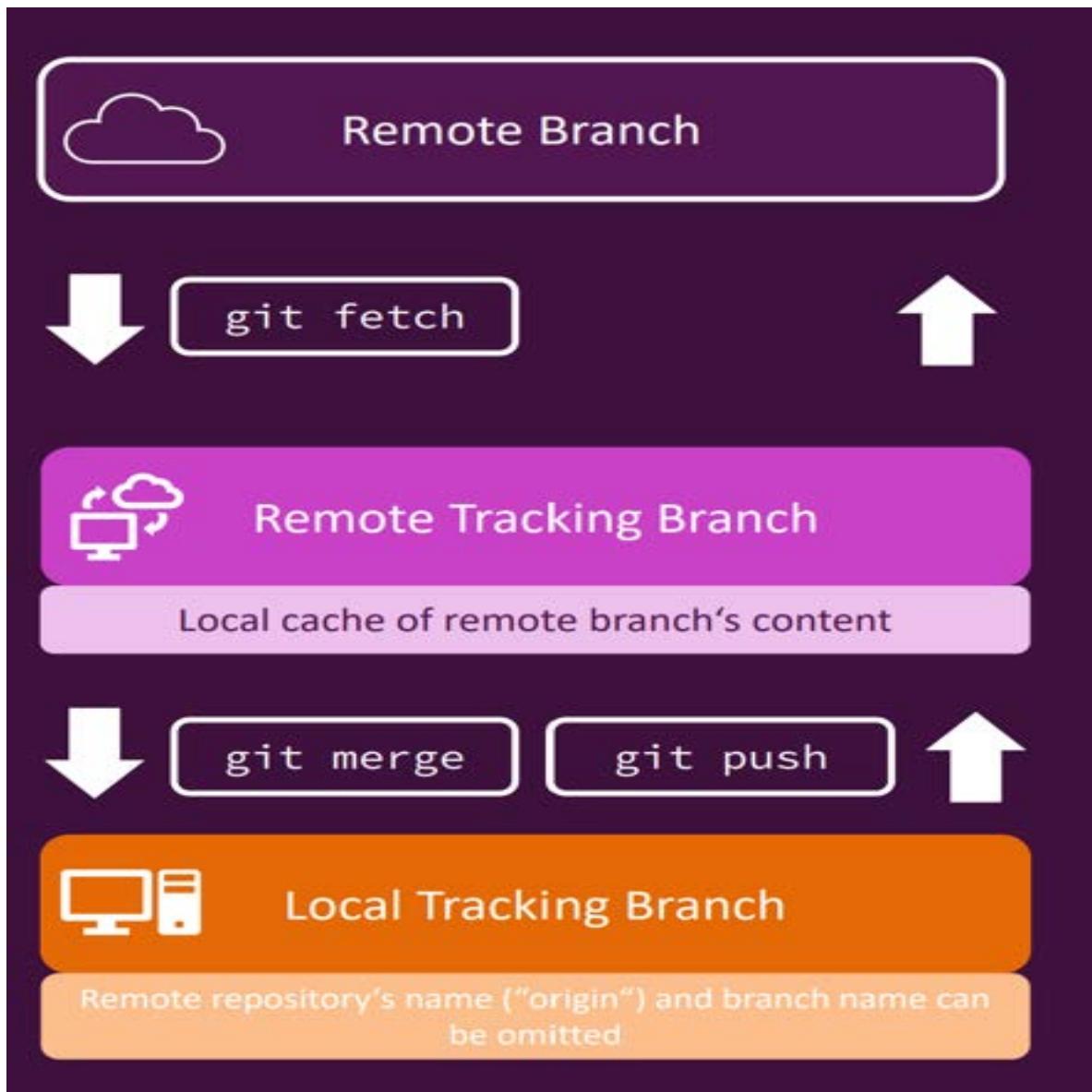


Fig. Local & remote tracking branches

Creating Local Tracking Branches:



```
D:\One Drive Data\Desktop\GitHub1-Basics>git branch -a
* (HEAD detached at cala248)
  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 cala248 remote 3
Switched to branch 'feature'

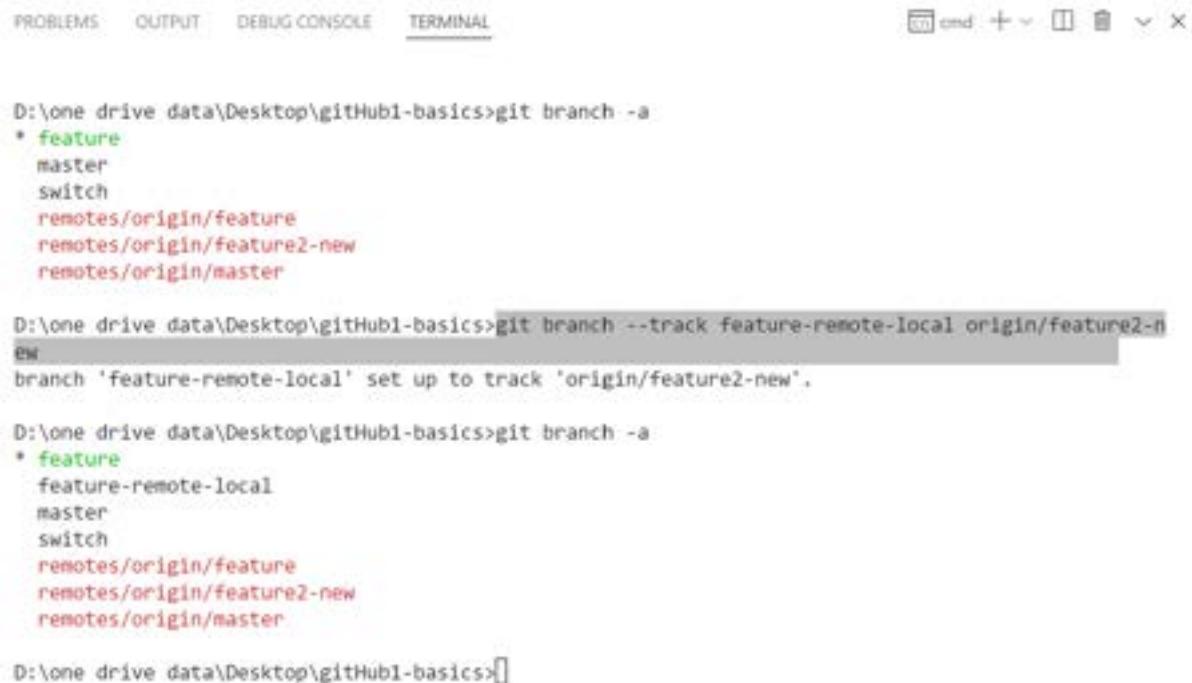
D:\One Drive Data\Desktop\GitHub1-Basics>git branch -a
* feature
  master
  switch
  remotes/origin/feature
  remotes/origin/feature2-new
  remotes/origin/master

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



```
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>git branch -a
'git' 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 m1.txt, remote.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>
- OUTLINE** and **TIMELINE** buttons are visible at the bottom of the sidebar.

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 m1.txt, remote.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
 caia248..e9e34c7 feature2-new -> feature2-new
 - D:\one drive data\Desktop\GitHub1-basics>
- OUTLINE** button is visible at the bottom of the sidebar.

Fig. Success on remote tracking branches

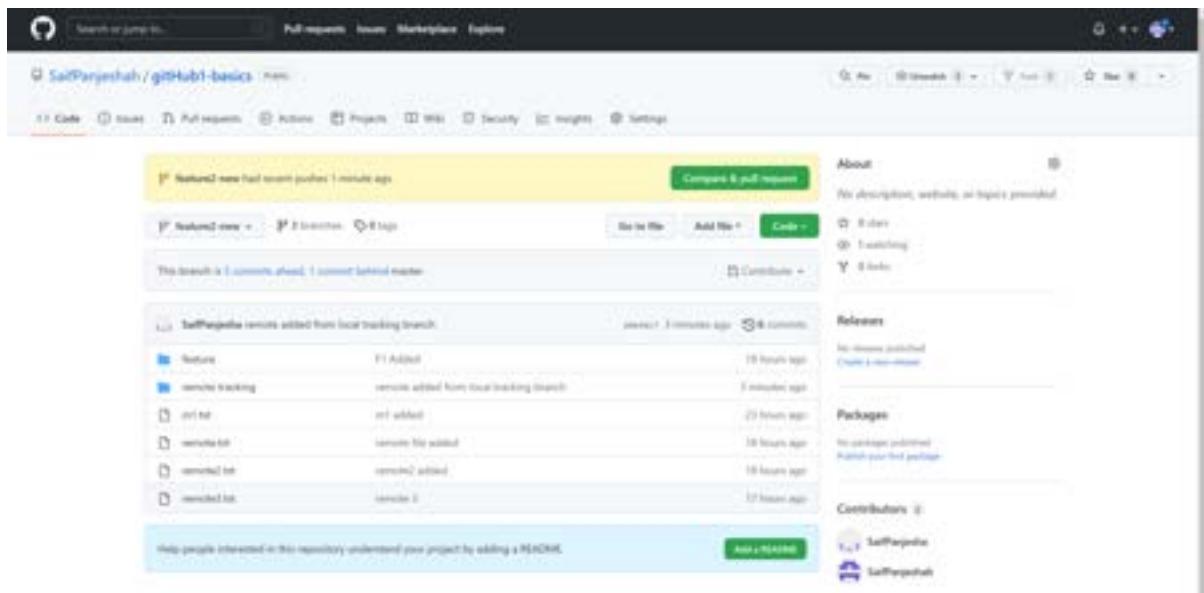


Fig. Successful on remote added from local tracking branches

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

Created New File :

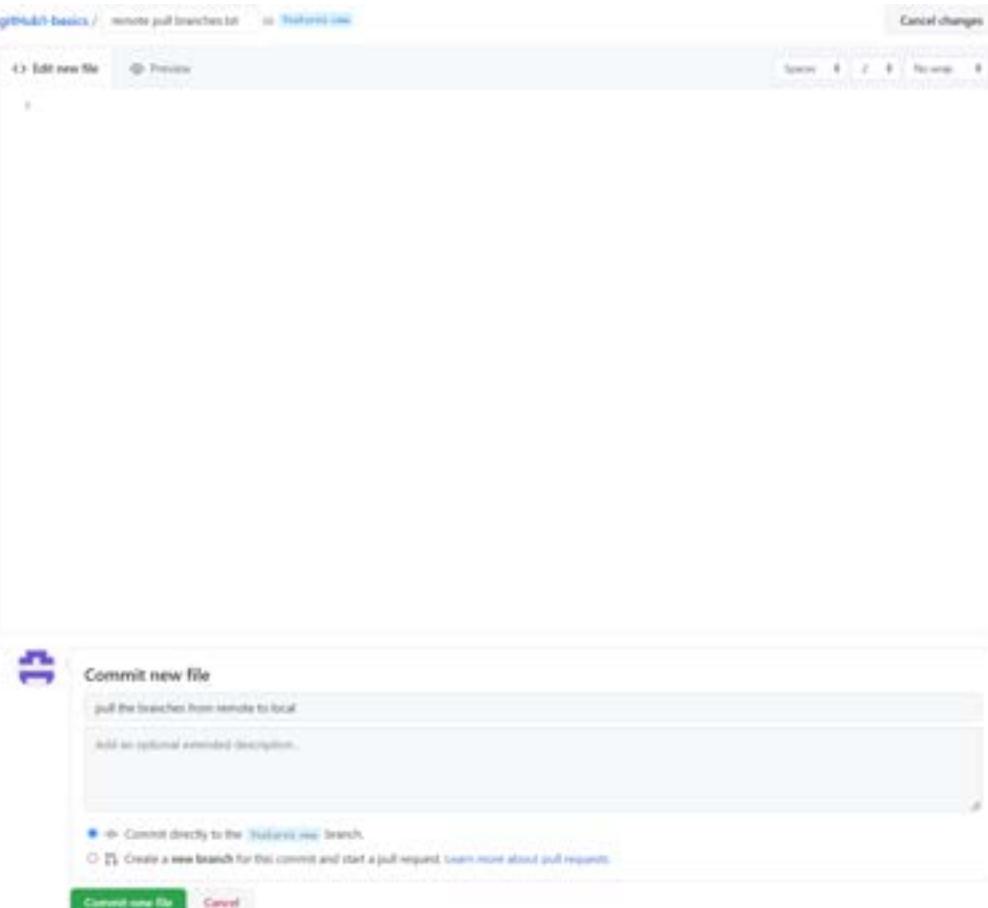


Fig. Created a new File on remote tracking branches

git branch -vv: List local tracking branches and their remotes

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

```
D:\One Drive\data\Desktop\github1-basics>git pull
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 2 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (2/2), 651 bytes | 198.00 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
```

Fig. Successful added to local tracking branches from remote vice versa

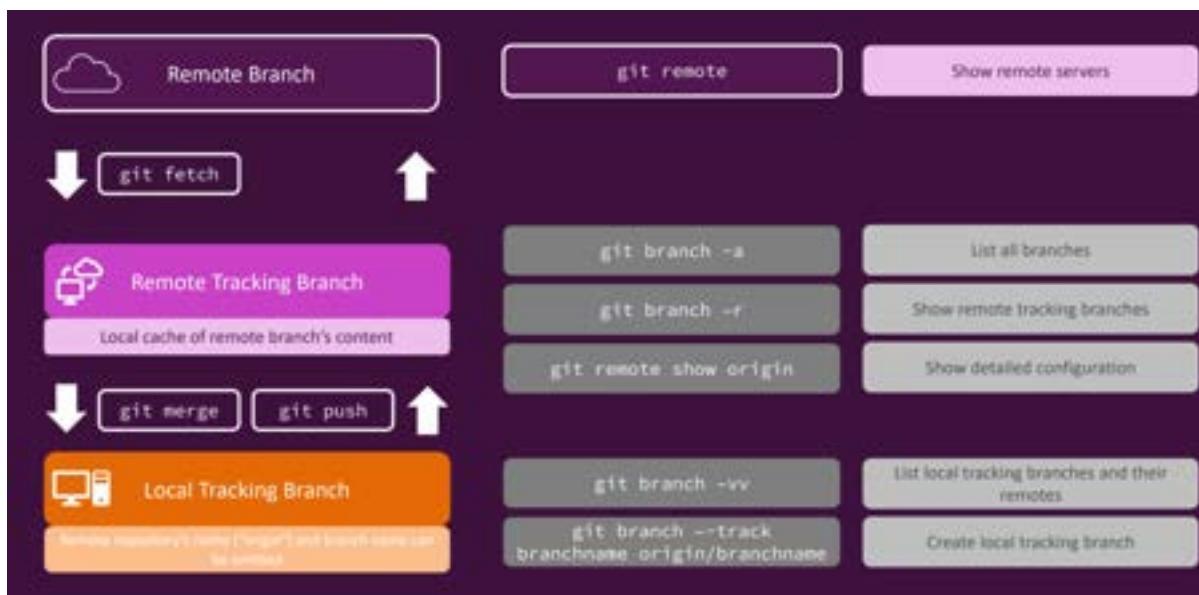


Fig. Local & Remote Tracking Branches Commands

Cloning a remote repository:

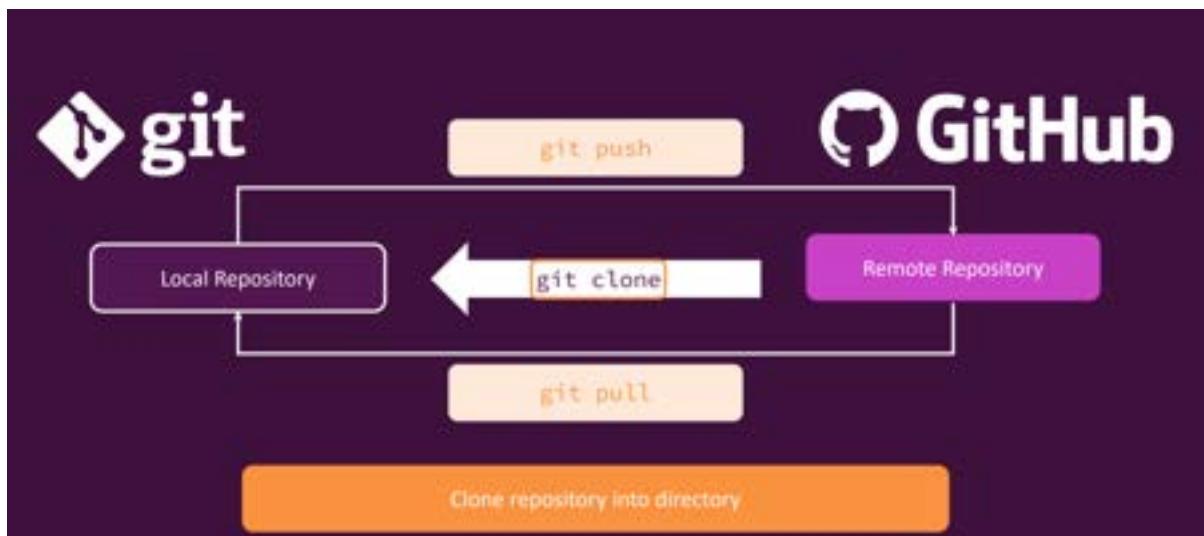


Fig. Shows How to Clone a remote repository

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

The screenshot shows a terminal window with the following output:

```
D:\One Drive\DATA\Desktop\clone>git clone https://github.com/SaifPanjeshah/gitHub1-basics.git
Cloning into 'gitHub1-basics'...
remote: Enumerating objects: 20, done.
remote: Counting objects: 100% (20/20), done.
remote: Compressing objects: 100% (13/13), done.
remote: Total 20 (delta 7), reused 9 (delta 2), pack-reused 0
Receiving objects: 100% (20/20), done.
Resolving deltas: 100% (7/7), done.

D:\One Drive\DATA\Desktop\clone>[]
```

Fig. Cloning repository into folder

```
D:\One Drive\DATA\Desktop\clone>git status
fatal: not a git repository (or any of the parent directories): .git
```

Fig. Fatal error while using git commands

How to resolve this fatal error?

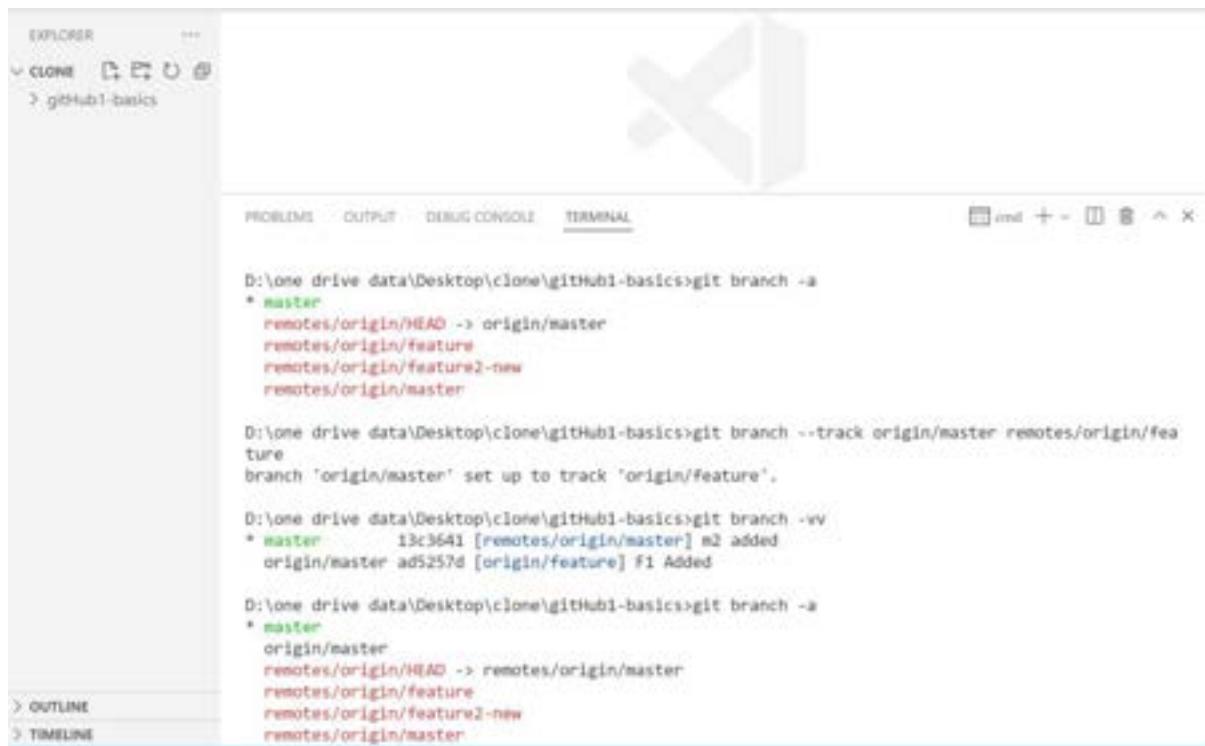


A screenshot of the Visual Studio Code interface. The Explorer sidebar on the left shows a folder named 'clone' containing a subfolder 'github1-basics'. The main area is a terminal window with the following text:

```
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



A screenshot of the Visual Studio Code interface. The Explorer sidebar on the left shows a folder named 'clone' containing a subfolder '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

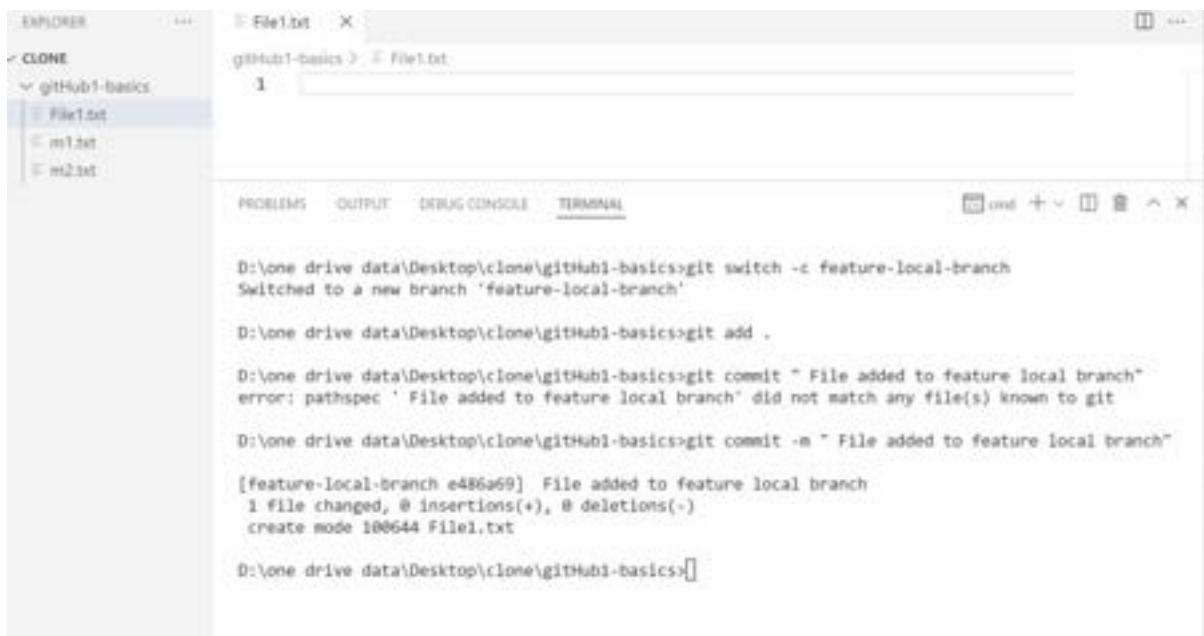
D:\one drive data\Desktop\clone\github1-basics>git branch --track origin/master remotes/origin/feature
branch 'origin/master' set up to track 'origin/feature'.

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 terminal window displays the following git commands and their output:

```
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 -m "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 terminal window displays the following git commands and their output:

```
D:\One Drive\DATA\Desktop\clone\github1-basics>git branch
* feature-local-branch
  master
  origin/master

D:\One Drive\DATA\Desktop\clone\github1-basics>git push origin feature-local-branch
fatal: credential-cache unavailable; no unix socket support
fatal: credential-cache unavailable; no unix socket support
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 282 bytes | 282.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'feature-local-branch' on GitHub by visiting:
remote:   https://github.com/SaifPanjeshah/github1-basics/pull/new/feature-local-branch
remote:
To https://github.com/SaifPanjeshah/github1-basics.git
 * [new branch]      feature-local-branch -> feature-local-branch

D:\One Drive\DATA\Desktop\clone\github1-basics>
```

Fig. Successful push on remote branch

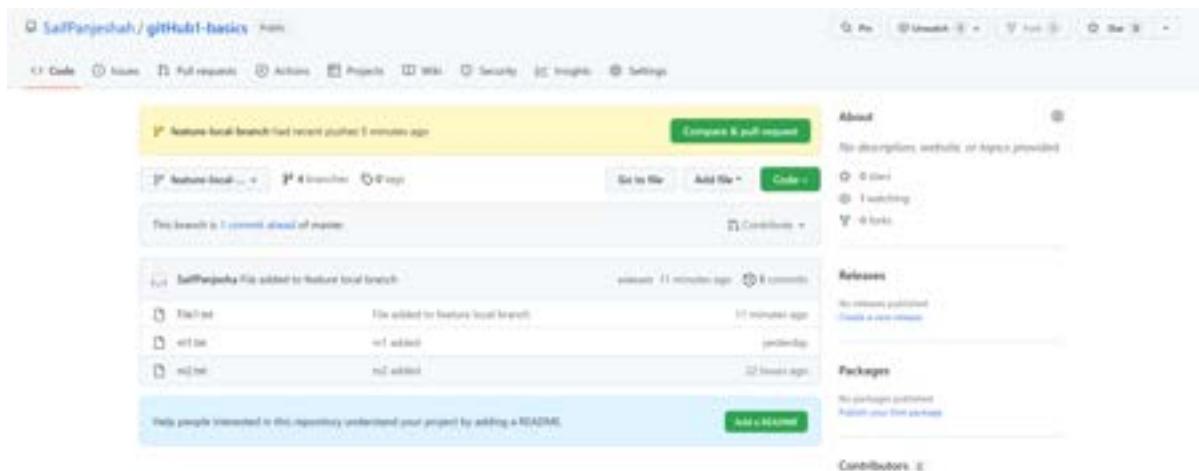


Fig. Successful added on remote branch GitHub

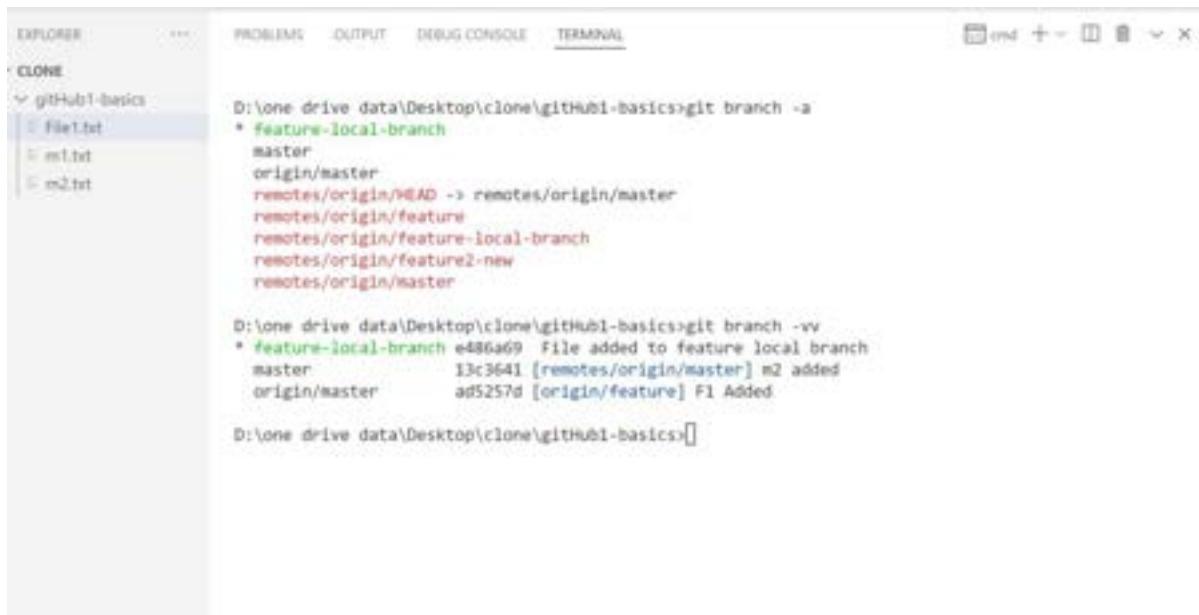


Fig. Shows all tracking branches

Now, our feature-local-branch refer to any kind of well remote tracking branch? So How Can we turn into local tracking branch?

```
D:\one drive data\Desktop\clone\gitHub1-basics>git branch -vv
* feature-local-branch e486a69 File added to feature local branch
  master      13c3641 [remotes/origin/master] m2 added
  origin/master ad5257d [origin/feature] F1 Added
```

```
D:\one drive data\Desktop\clone\gitHub1-basics> git switch master
Switched to branch 'master'
Your branch is up to date with 'remotes/origin/master'.
```

```
D:\one drive data\Desktop\clone\gitHub1-basics>git branch -D feature-local-branch
Deleted branch feature-local-branch (was e486a69).
```

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

The screenshot shows the VS Code interface with the 'TERMINAL' tab selected. In the terminal, the user runs several commands:

- `git branch -a` shows branches: * master, origin/master, remotes/origin/HEAD -> remotes/origin/master, remotes/origin/feature, remotes/origin/feature-local-branch, remotes/origin/feature2-new, remotes/origin/master.
- `git branch --track feature-local-branch origin/feature-local-branch` creates a local branch 'feature-local-branch' tracking 'origin/feature-local-branch'. The output says 'branch 'feature-local-branch' set up to track 'origin/feature-local-branch'.'
- `git branch -v` shows branch details:
 - * master 13c3641 [remotes/origin/master] m2 added
 - origin/master ad5257d [origin/feature] F1 Added
- `git status` shows the repository status.

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:

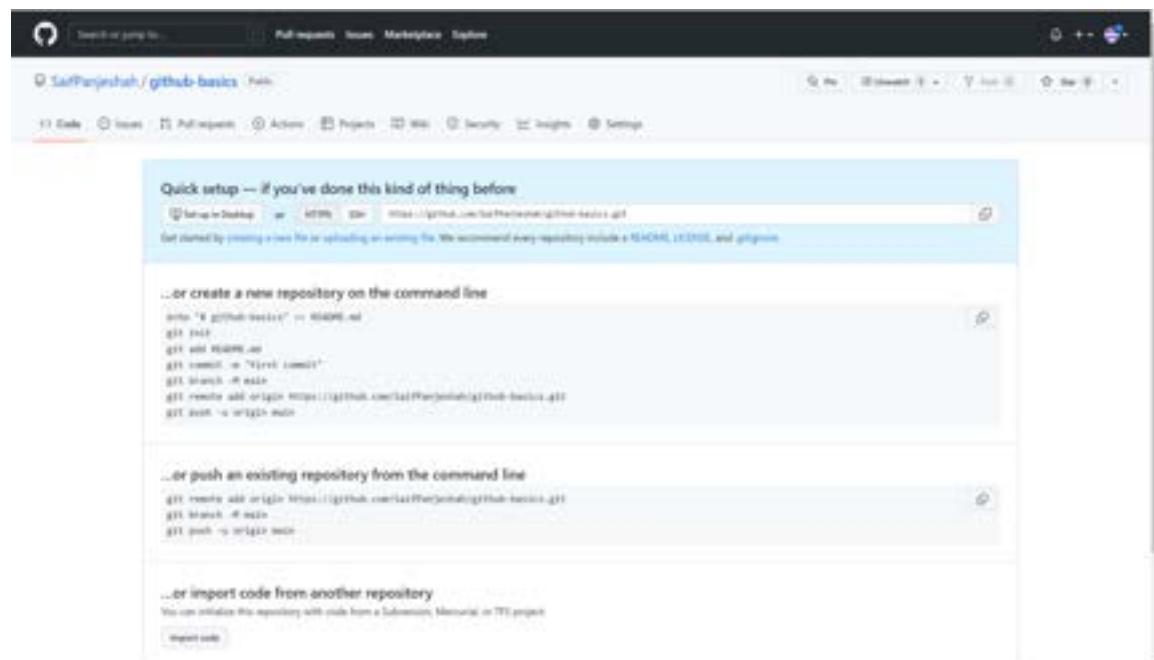


Fig. Checkout for “-u” Upstream

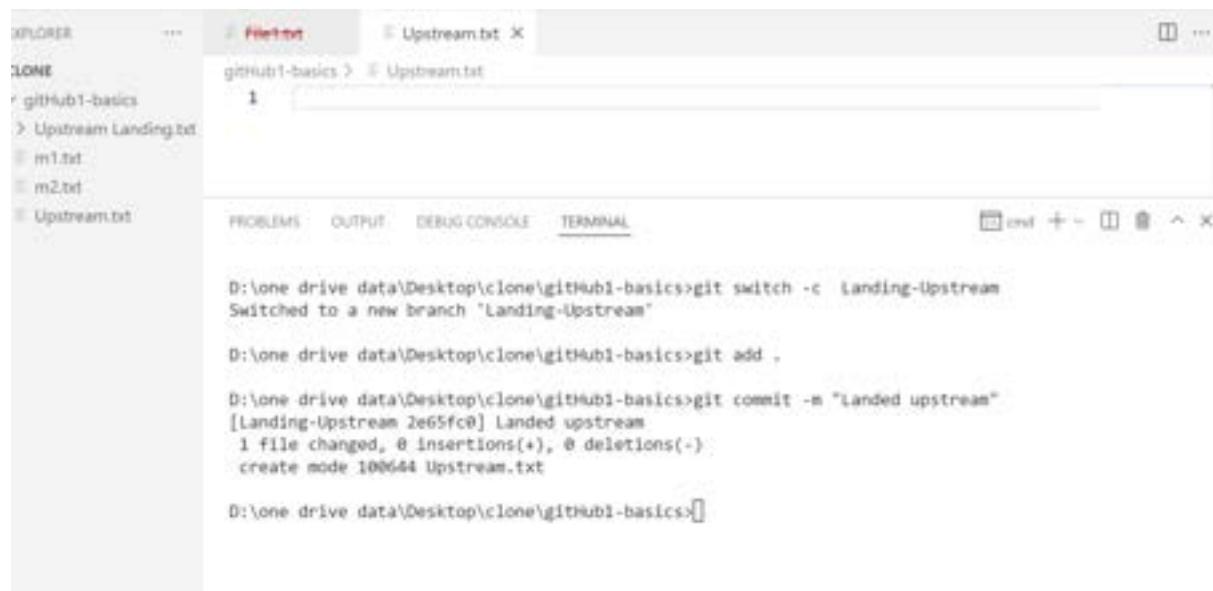


Fig. Created upstream branch

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

```
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/SalifPanjeshah/gitHub1-basics/pull/new/Landing-Upstream
remote:
To https://github.com/SalifPanjeshah/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 e486a09 [origin/feature-local-branch] File added to feature local branch
  master          13c3641 [remotes/origin/master] m2 added
  origin/master    ad5257d [origin/feature] F1 Added

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

Fig. Successful added on remote without tracking the branches

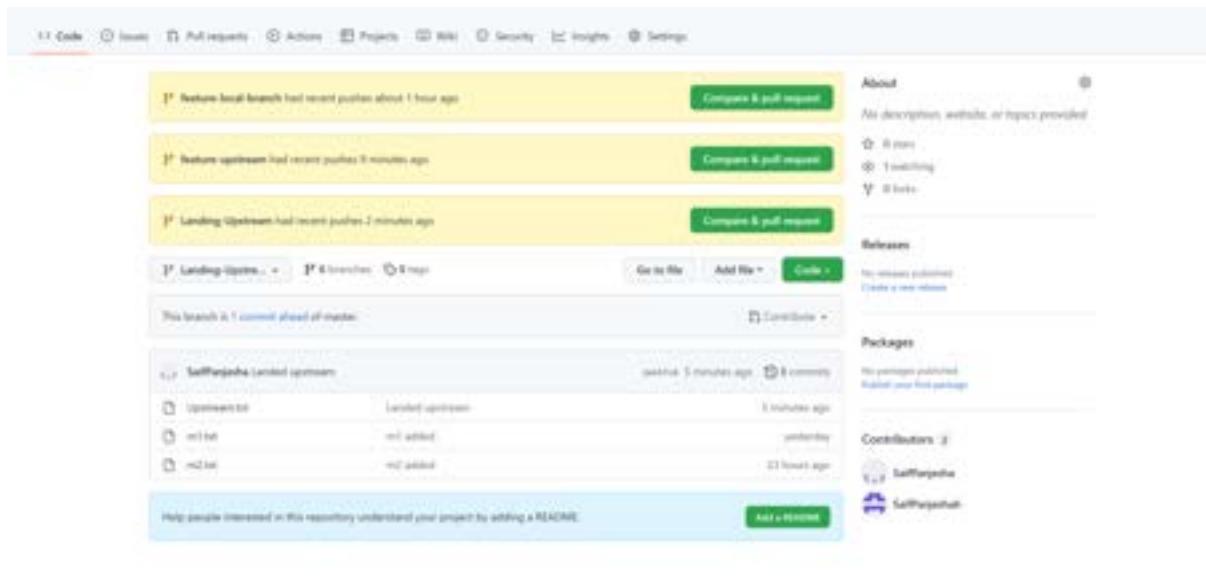


Fig. Successful Upstream.txt on GitHub

Deleting Remote Branches & Public Commits:

The screenshot shows the VS Code interface with the terminal tab active. The terminal output shows the following commands and their results:

```
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 following commands and their results:

```
D:\One Drive\DATA\Desktop\Clone\GitHub1-Basics>git branch -a
* Landing-Upstream
  feature-local-branch
  master
  origin/master
  remotes/origin/HEAD -> remotes/origin/master
  remotes/origin/Landing-Upstream
  remotes/origin/feature
  remotes/origin/feature-local-branch
  remotes/origin/feature-upstream
  remotes/origin/feature2-new
  remotes/origin/master

D:\One Drive\DATA\Desktop\Clone\GitHub1-Basics>git branch --delete --remote origin/feature-upstream
Deleted remote-tracking branch origin/feature-upstream (was 26360c7).

D:\One Drive\DATA\Desktop\Clone\GitHub1-Basics>
```

Fig. Deleted branches

The screenshot shows the VS Code interface with the terminal tab selected. The command 'git ls-remote' is run against the remote repository 'https://github.com/SaifPanjeshah/gitHub1-basics'. The output lists several branches and their corresponding remote references:

```
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
26368c7c2beec07260b27c0f3aed384738aa6b2d      refs/heads/feature-upstream
9e3af2165cc5f29e149eeeaa1207a96d2db77004b      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
26368c7c2beec07260b27c0f3aed384738aa6b2d      refs/heads/feature-upstream
9e3af2165cc5f29e149eeeaa1207a96d2db77004b      refs/heads/feature2-new
13c36418a467824c7c0d92cf5883dc83277fb22c      refs/heads/master
```

Fig. Successful deleted

The screenshot shows a GitHub repository page for 'SaifPanjehah/github-basics'. The 'Overview' tab is selected. The 'Default branch' section shows the 'main' branch. The 'Your branches' section lists several branches: 'Landing-branch', 'Feature-local-branch', 'Feature-new', and 'Feature'. The 'Active branches' section also lists these four branches. Each branch entry includes a commit link, a timestamp, and a 'New pull request' button.

Fig. feature-upstream successful deleted on GitHub

How to delete Public Commits:

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

```
D:\One Drive Data\Desktop\clone\github-basics>git switch master
Switched to branch 'master'
Your branch is up to date with 'remotes/origin/master'.

D:\One Drive Data\Desktop\clone\github-basics>git log
commit 13c36418a67824c7c0d92cf5883dc83277fb22c (HEAD -> master, origin/master, origin/HEAD)
Author: Saif Panjehah <98874394+SaifPanjehah@users.noreply.github.com>
Date:  Sun May 22 21:08:49 2022 +0530

    m2 added

commit 97cc11cd8342a5c23a854b14db17f9e8eb61c76
Author: Saif Panjehah <98874394+SaifPanjehah@users.noreply.github.com>
Date:  Sun May 22 18:23:33 2022 +0530

    m1 added

D:\One Drive Data\Desktop\clone\github-basics>
```

Fig. Public Commits

```
D:\one drive data\Desktop\clone\gitHub1-basics>git log
commit 13c36418a467824c7c8d92cf5883dc83277fb22c (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 97cc11cd8342a5c23a854b14db17f9e8eb61c76
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

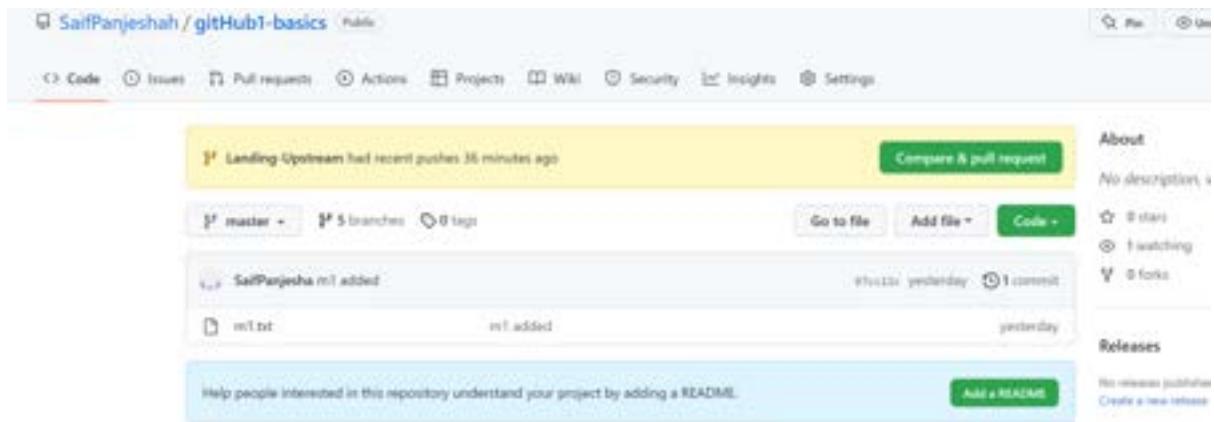


Fig. Success on Deletion GitHub

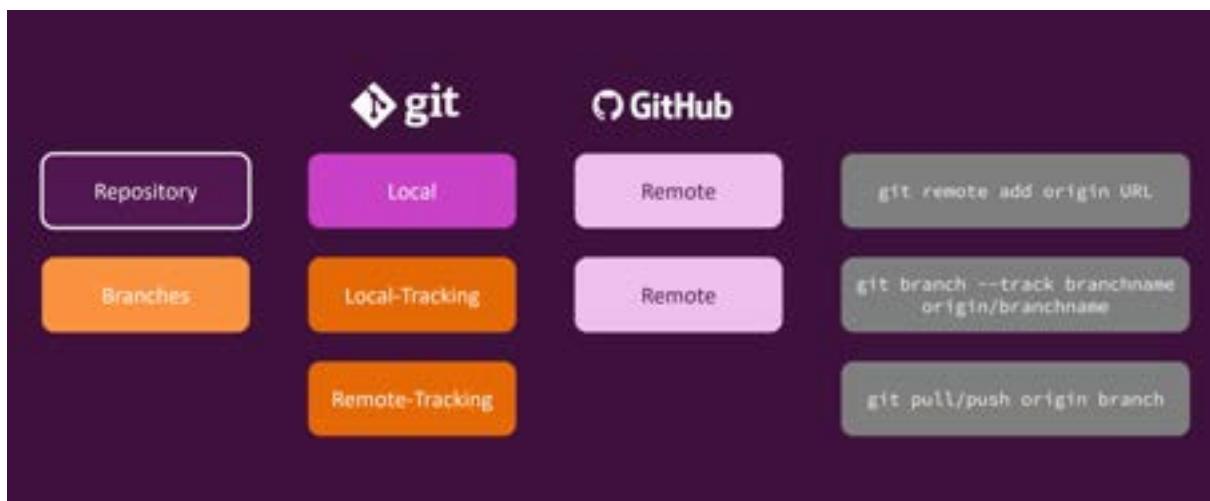


Fig. Wrap up summary

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:

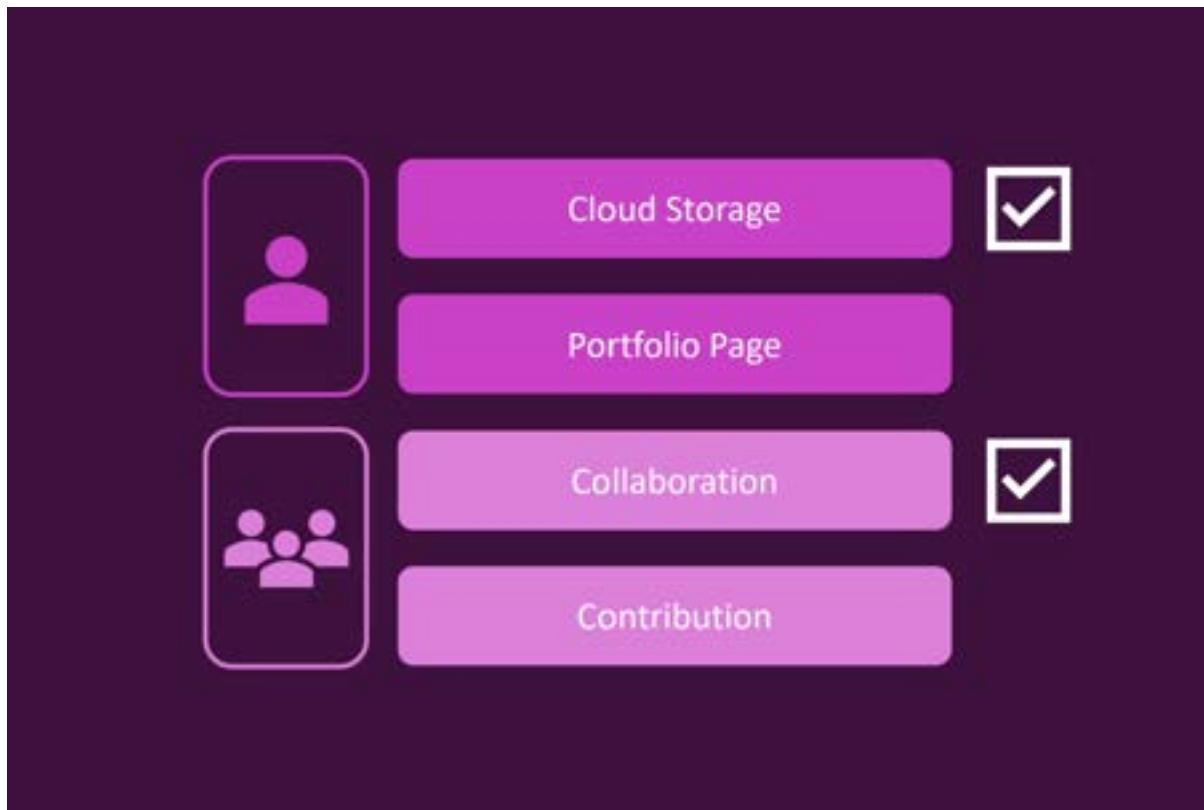


Fig. The 4 GitHub Use Cases

- **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>

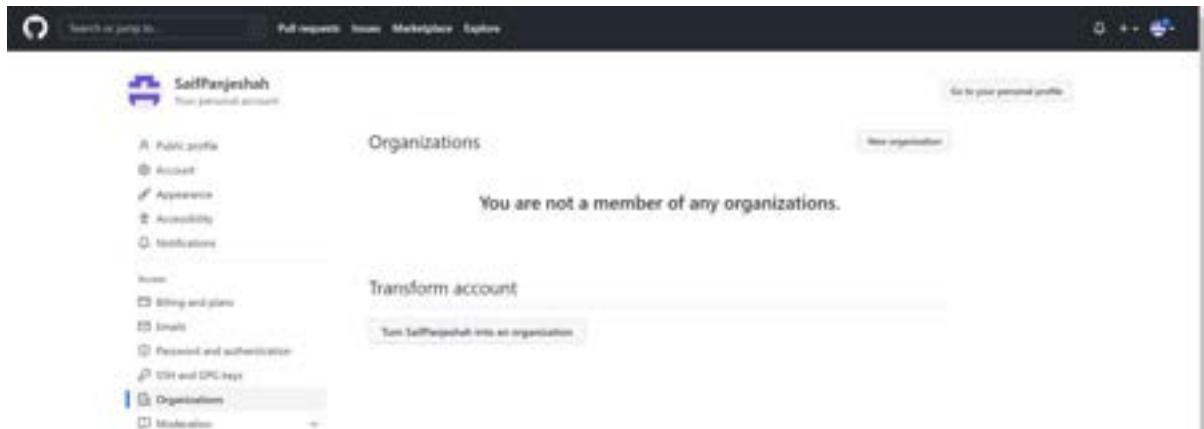


Fig. Personal GitHub Account

Changing Repository Type from Public to Private:

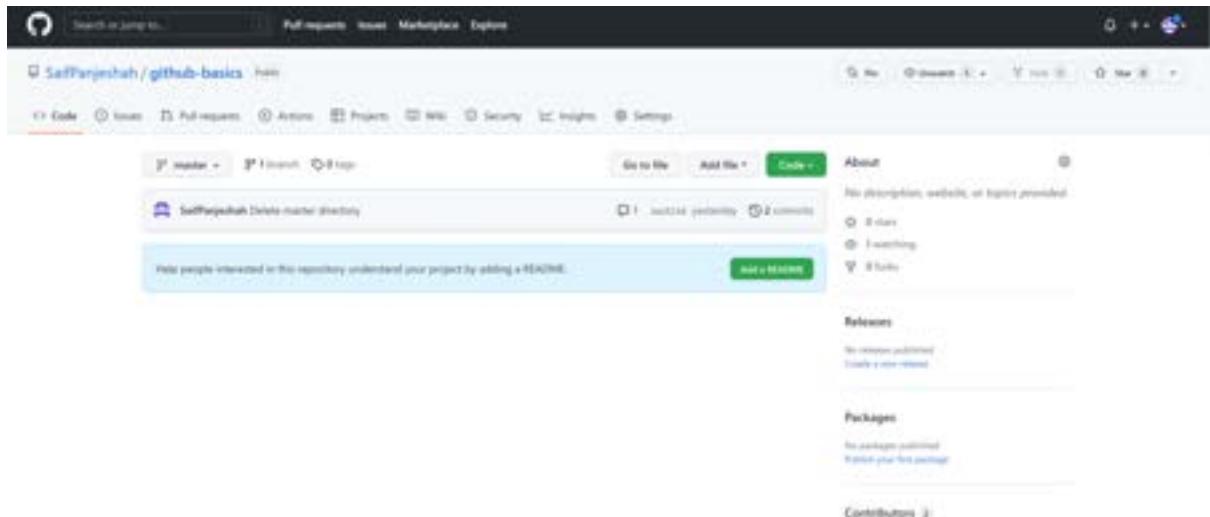


Fig. Shows Repository added in our account

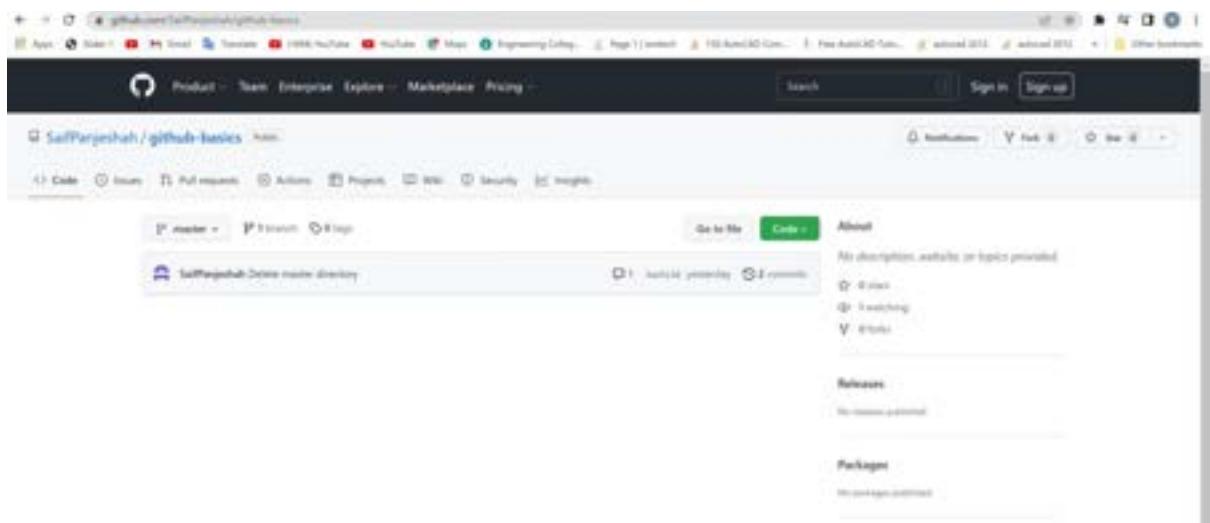


Fig. Accessing repositories from outside the browser or any user

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

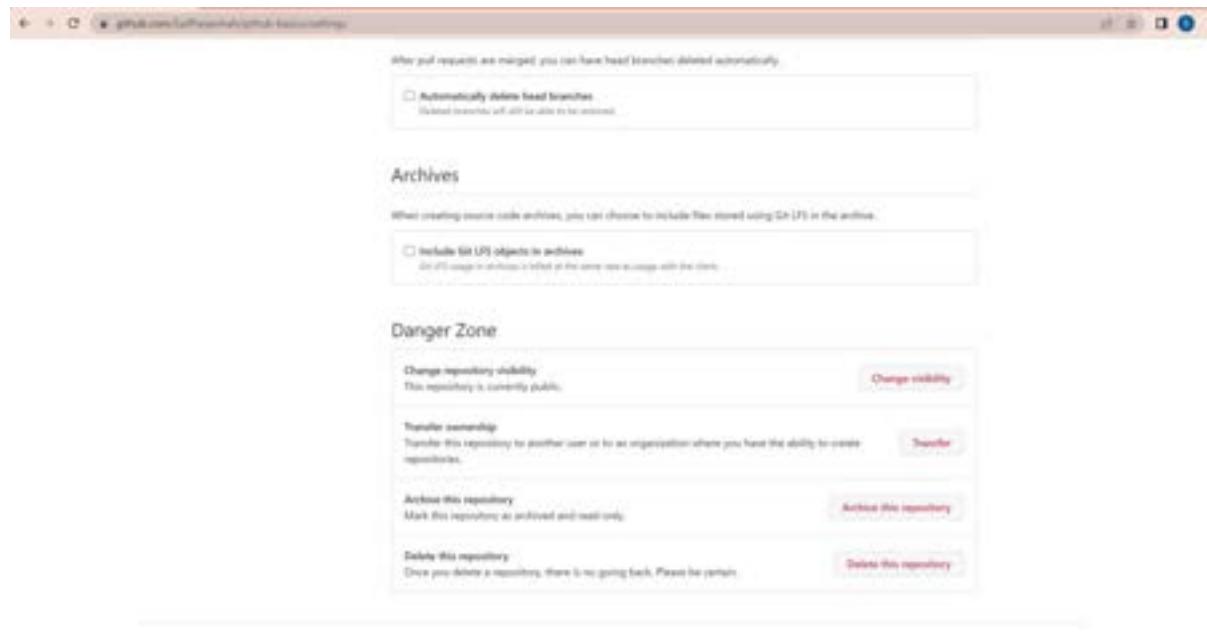


Fig. Visibility Setting of repositories

Automatically delete head branches

Delete

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.](#)

Archives

When creating a new commit, include LFS in the archive.

Danger

Change visibility

Transfer ownership

Archive this repository

Delete this repository

Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About



Confirm access

Password

.....

[Forgot password?](#)

[Confirm password](#)

Tip: You are entering [sudo mode](#). We won't ask for your password again for a few hours.

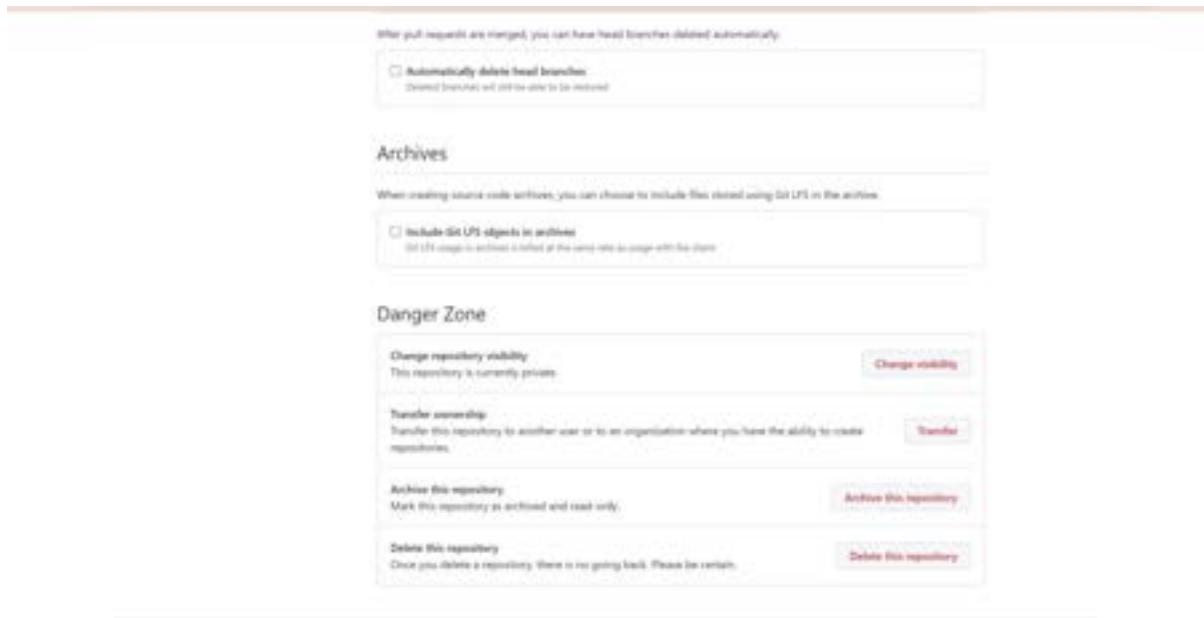


Fig. Successful Changes from public to private repository

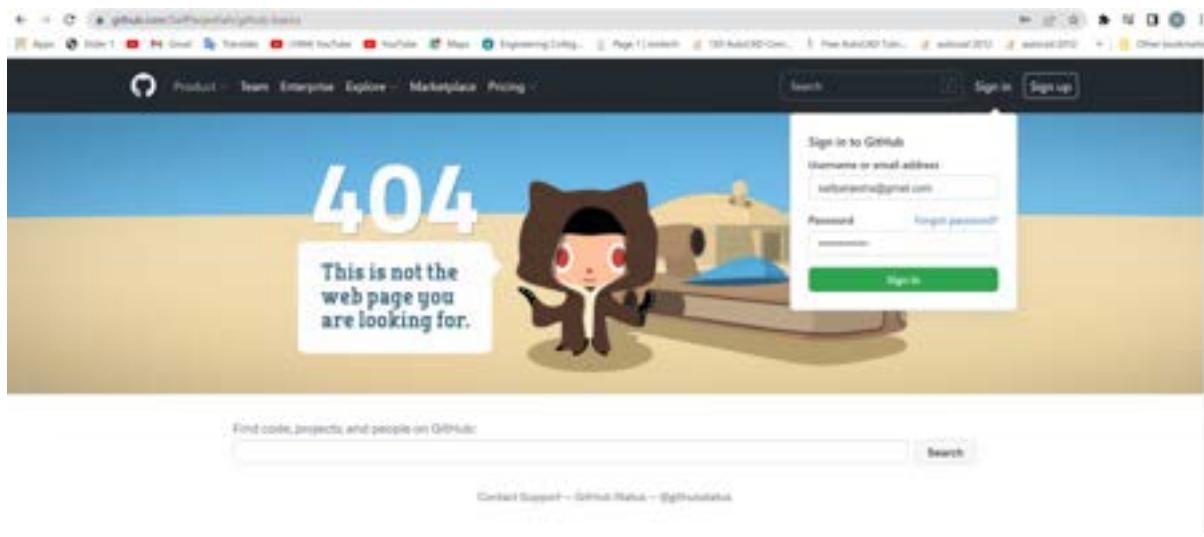


Fig. No one can access the repository

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

Pushing Commits to Public Repository

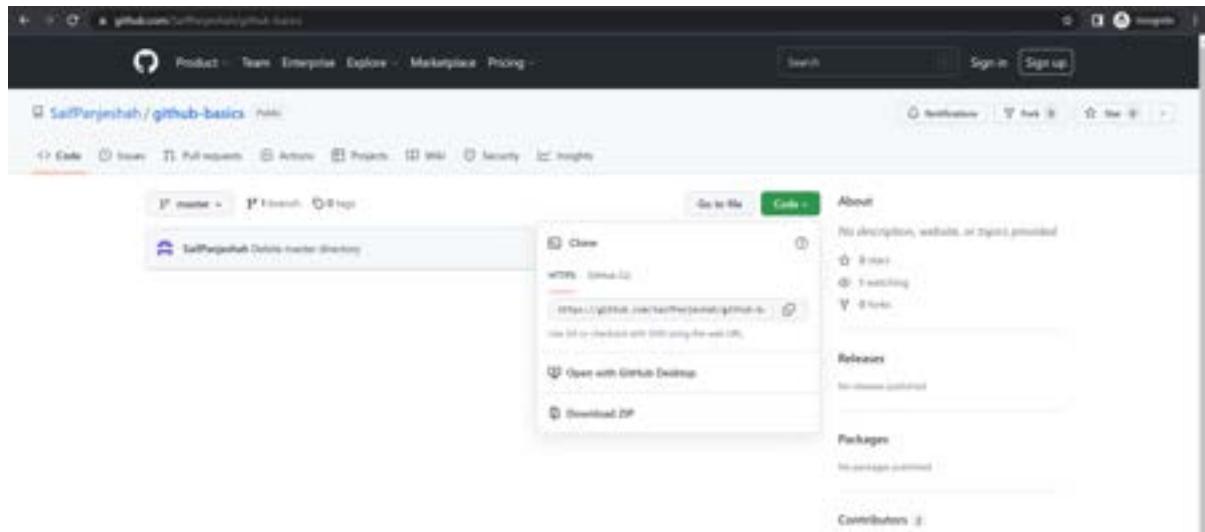


Fig. Changing the repository to public

Copying the Http URL: <https://github.com/SaifPanjeshah/github-basics.git> and create new folder in vs code push the information our actual GitHub Project



The screenshot shows the VS Code interface. In the Explorer sidebar, there is a folder named "github-basics". The terminal tab is active, displaying the following command-line session:

```
D:\One Drive\DATA\Desktop\push commits to public repo>git clone https://github.com/SaifPanjeshah/github-basics.git
Cloning into 'github-basics'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 6 (delta 0), reused 4 (delta 0), pack-reused 0
Receiving objects: 100% (6/6), done.

D:\One Drive\DATA\Desktop\push commits to public repo>cd github-basics

D:\One Drive\DATA\Desktop\push commits to public repo\github-basics>git branch
* master

D:\One Drive\DATA\Desktop\push commits to public repo\github-basics>
```

Below the terminal, the Outline and Timeline tabs are visible.

Fig. Created new push commits to public folder

Windows Credentials	Add a Windows credential
KRJ	Modified: 11/04/2022
Certificate-Based Credentials	Add a certificate-based credential
No certificates.	
Generic Credentials	Add a generic credential
MSK-Skype for Desktop MSAv2\ivesaifazalpanjesh...	Modified: Today
MSK-Skype for Desktop\ivesaifazalpanjesh101	Modified: Today
vscode\vscode.github-authentication\github.auth	Modified: 22/05/2022
MicrosoftAccount\user\saffazalpanjesh101@gmail.c...	Modified: Today
MicrosoftAccount\user\saffazalpanjesh1@outlook.c...	Modified: Today
OneDrive Cached Credential	Modified: Today
virtualapp\digitalical	Modified: 30/04/2022

Fig. deleting all GitHub credentials

The screenshot shows the GitHub developer settings page under 'Personal access tokens'. It displays a list of tokens with their names and creation dates. One token, 'push-to-public', has a red circle with a minus sign over it, indicating it has been deleted. A note below the table states: 'Personal access tokens function like primary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate code in the API via Basic Authentication.'

Fig. deleted personal access tokens

The screenshot shows a VS Code interface with a terminal window open. The terminal output shows several attempts to push changes to a GitHub repository named 'github-basics'. The first two attempts succeed, but the third attempt fails with the error 'fatal: credential-cache unavailable; no unix socket support'. The fourth attempt also fails with the same error. The fifth attempt succeeds, pushing the changes to the 'master' branch.

```
D:\One Drive Data\Desktop\push commits to public repo\github-basics>git add .
D:\One drive data\Desktop\push commits to public repo\github-basics>git commit -m "pushes files on
to commits to public repo"
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean

D:\One drive data\Desktop\push commits to public repo\github-basics>git push
fatal: credential-cache unavailable; no unix socket support
fatal: credential-cache unavailable; no unix socket support
Everything up-to-date

D:\One drive data\Desktop\push commits to public repo\github-basics>git push origin master
fatal: credential-cache unavailable; no unix socket support
fatal: credential-cache unavailable; no unix socket support
Everything up-to-date

D:\One drive data\Desktop\push commits to public repo\github-basics>
```

Fig. Shows with access other users can't push the information our actual GitHub

How GitHub Account Manager Security:



Fig. Added Account Manager Security

Understanding & Adding Collaborator to a Private User Accounts:

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

- Explorer** sidebar: Shows a folder named "push-commits-to-public" containing "github-basic" and "push-to-public" branches. The "push1.txt" file is selected.
- Terminal** tab: Displays the command-line output of a git push attempt.

```
B:\One Drive Data\Desktop\push commits to public repo\github-basic>git add .
B:\One Drive Data\Desktop\push commits to public repo\github-basic>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

B:\One drive data\Desktop\push commits to public repo\github-basic>git push
fatal: credential-cache unavailable; no unix socket support
fatal: credential-cache unavailable; no unix socket support
Everything up-to-date

B:\One drive data\Desktop\push commits to public repo\github-basic>git push origin master
fatal: credential-cache unavailable; no unix socket support
fatal: credential-cache unavailable; no unix socket support
Everything up-to-date

B:\One drive data\Desktop\push commits to public repo\github-basic>
```

Fig. Shows with access other users can't push the information our actual GitHub

To resolve this problem:

The screenshot shows the 'Settings' page for a repository named 'SaifPanjehshah/github-basics'. The 'General' tab is selected. Under 'Who has access', there are two sections: 'PUBLIC REPOSITORY' (radio button selected) and 'DIRECT ACCESS' (radio button unselected). The 'PUBLIC REPOSITORY' section states: 'This repository is public and visible to anyone.' and has a 'Manage' link. The 'DIRECT ACCESS' section states: '0 collaborators have access to this repository. Only you can contribute to this repository.' and also has a 'Manage' link. On the left sidebar, under 'Access', 'Collaborators' is highlighted. Other sections include 'Code and automation', 'Security', and 'Integrations'.

Fig. Adding Collaborator to a Private User Accounts

The screenshot shows the 'Settings' page for the same repository. Under 'Who has access', the 'DIRECT ACCESS' section is selected, showing '1 has access to this repository. 0 collaborators, 1 invitation.' Below this, the 'Manage access' section shows a table with one row. The row contains a checkbox, a user icon, the name 'SaifPanjehshah', the status 'Awaiting SaifPanjehshah's response', a 'Pending invite' checkbox, and a 'Remove' link. A green 'Add people' button is located at the top right of the 'Manage access' section. Navigation links 'Previous' and 'Next >' are at the bottom.

Fig. Pending User Invitation

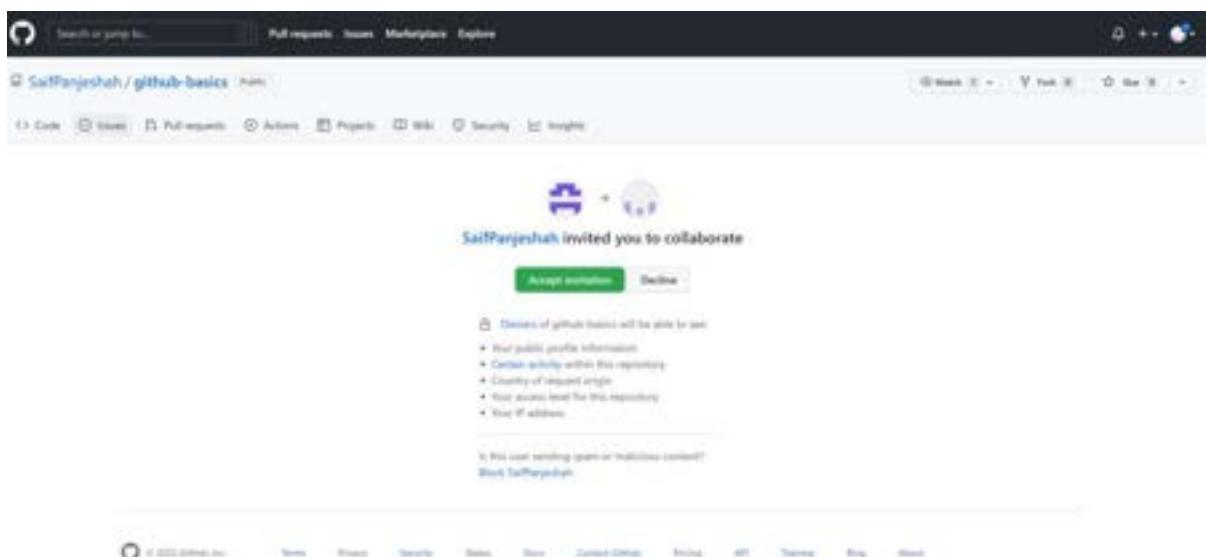


Fig. Accepting User request from another user account

A screenshot of the 'Who has access' section in a GitHub repository settings. It shows two main sections: 'PUBLIC REPOSITORY' and 'DIRECT ACCESS'. The 'PUBLIC REPOSITORY' section indicates that the repository is public and visible to anyone, with a 'Manage' link. The 'DIRECT ACCESS' section shows that 1 person has access to the repository, specifically 'SaifPanjeshah' (Collaborator), with a 'Remove' link. Below this, there's a 'Manage access' section with a 'Select all' checkbox, a search bar ('Find a collaborator...'), and a table listing the collaborator. The table includes columns for a checkbox, the collaborator's profile picture, the name 'SaifPanjeshah', the role 'Collaborator', and a 'Remove' link. Navigation links for 'Previous' and 'Next' are at the bottom.

Fig. Successful added

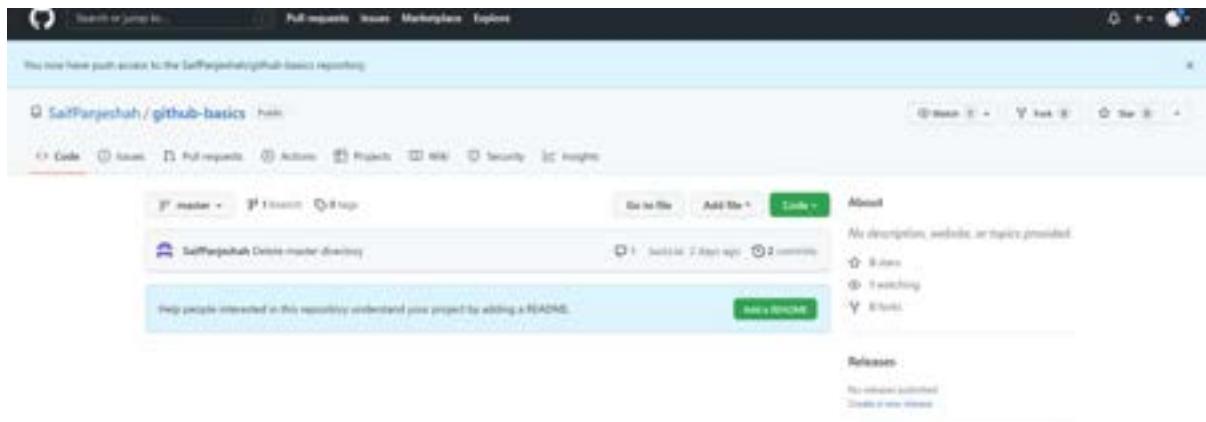


Fig. push access to another users

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)

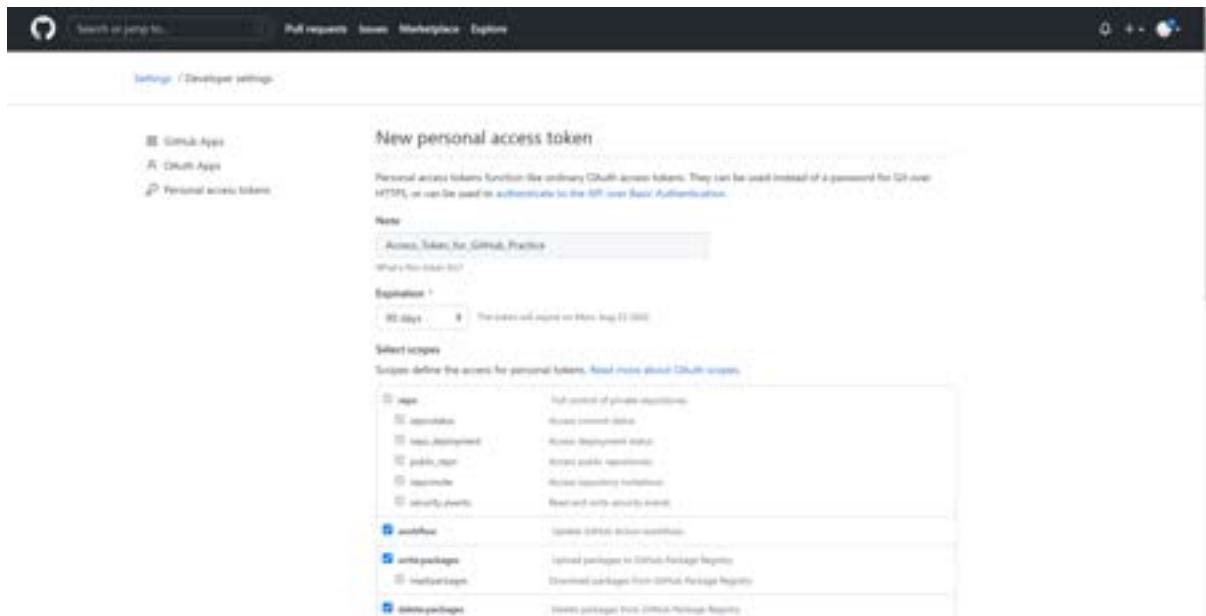


Fig. creating access Token From another user accounts

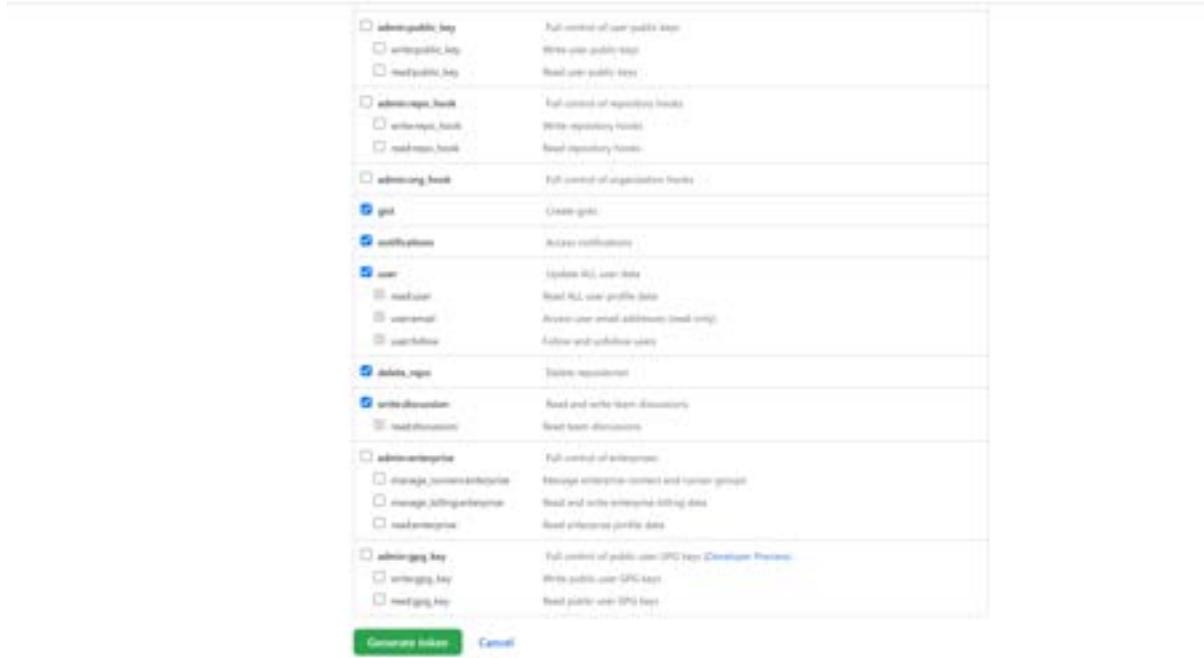


Fig. Creation Successful generate the token now without admin access

Created new file1.txt from local tracking branch of another user

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

```

D:\One Drive Data\Desktop\push commits to public repo\github-basics>git add .

D:\One Drive Data\Desktop\push commits to public repo\github-basics>git commit -m "Successful push to public"
[master 67@elcb] Successful push to public
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file1.txt

D:\One Drive Data\Desktop\push commits to public repo\github-basics>git push
fatal: credential-cache unavailable; no unix socket support
fatal: credential-cache unavailable; no unix socket support
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Writing objects: 100% (3/3), 273 bytes | 273.00 KiB/s, done.
Total 3 (delta 0), reused 1 (delta 0), pack-reused 0
To https://github.com/SaifPanjeshah/github-basics.git
 3a2513d..67@elcb master -> master

```

Fig. Succesful push the changes in account

To view please check the owner GitHub account

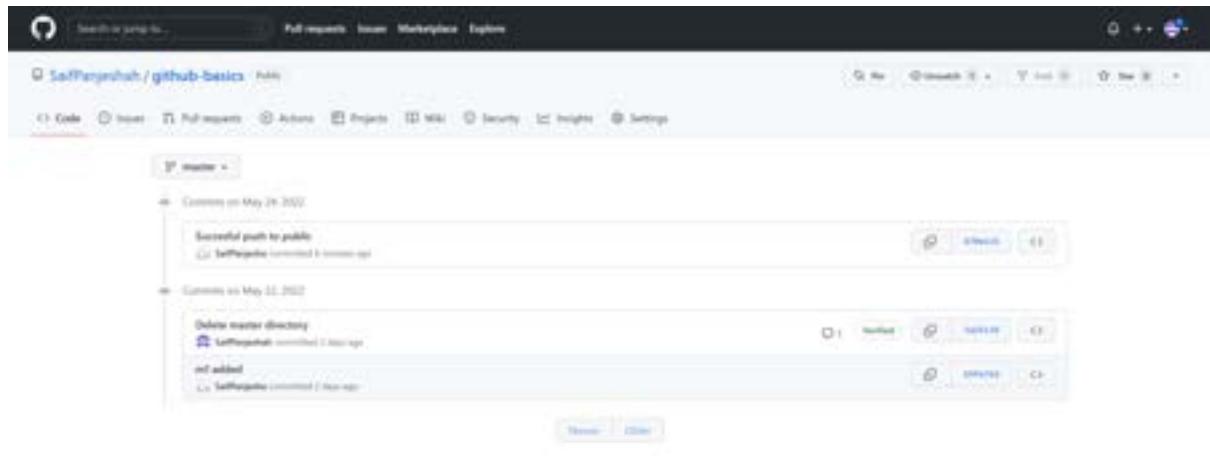


Fig. Commit Succesful added

Collaborating in Private repositories:

Danger Zone



Fig. Changing repositories in private

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

- Explorer:** Shows a tree view with a expanded node "PUSH COMMITS TO PUBLIC". Inside it, "push-to-public" is expanded, showing "push1.txt". Other files like "file1.txt", "file2.txt", and ".gitignore" are also listed.
- Terminal:** Displays the command-line output of a git push operation:

```
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 repo"
[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>
```
- Bottom Navigation:** Shows "OUTLINE" and "TIMELINE" buttons.

Fig. Successful push to private repository

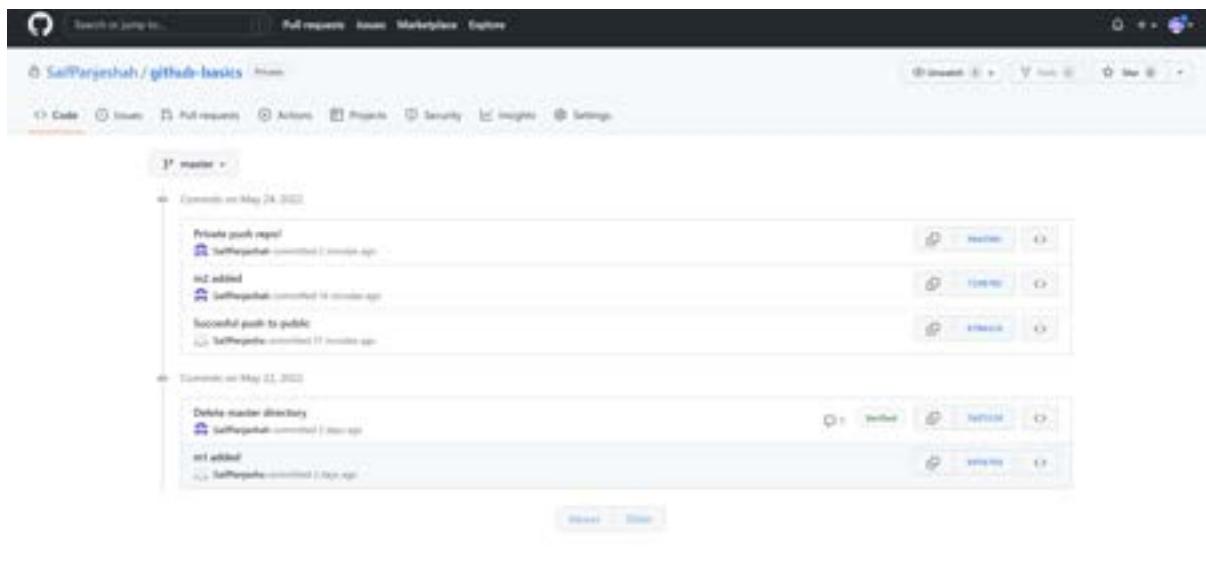


Fig. Successful push to private repository in GitHub

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**

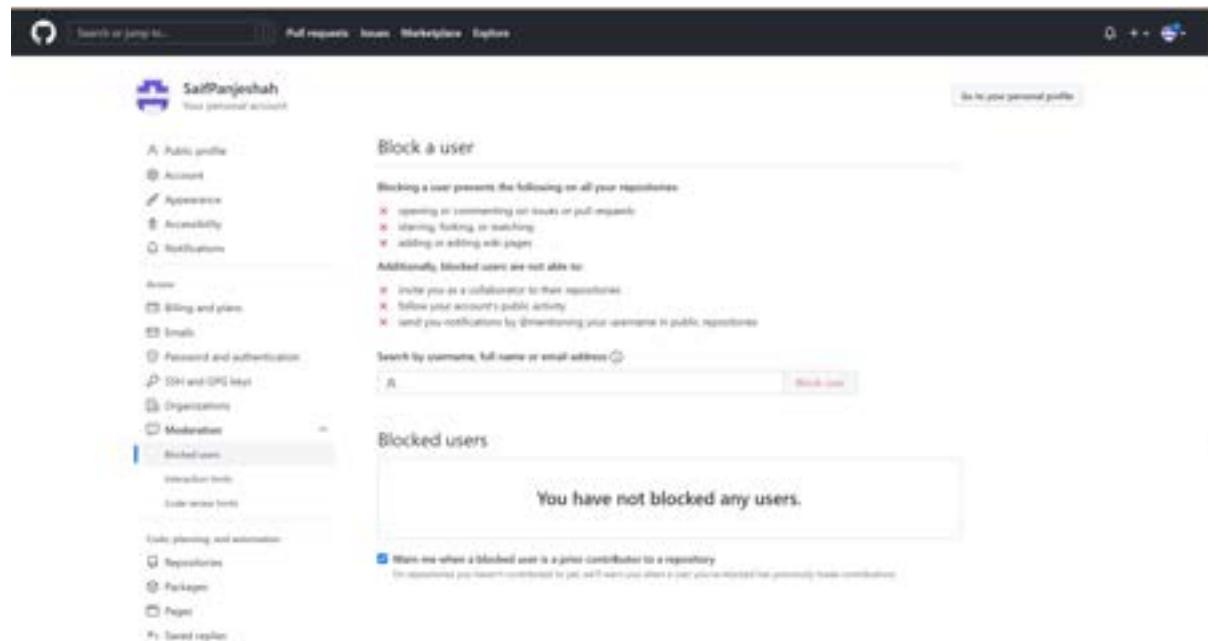


Fig. Block a user's

The screenshot shows the 'Temporary interaction limits' section of the GitHub account settings. On the left, there's a sidebar with various account management options like Public profile, Account, Appearance, Accessibility, Notifications, Access, Billing and plans, Emails, Password and authentication, SSH and GPG keys, Organizations, Moderation, Shared repos, Interaction limits, Code review limits, Code planning and automation, Repositories, Packages, and Pages. The 'Interaction limits' option is currently selected. The main content area is titled 'Temporary interaction limits' and contains three sections: 'Limit for existing users', 'Limit for prior contributors', and 'Limit for repository collaborators'. Each section has a 'Enable' button and specific configuration options for 'New users', 'New commits', 'Contributions', and 'Collaborations'.

Fig. Limit Iteration

The screenshot shows the 'General' tab of a GitHub repository settings page for 'Safipanjeshah / github-basics'. The repository name is 'github-basics'. The 'General' tab includes sections for 'About', 'Re: Contributors', 'Moderation options', 'Code and automation', 'Topics', 'Actions', 'Webhooks', and 'Environments'. The 'Social preview' section allows for uploading a custom image for social media sharing, with a note that the image should be at least 640x320px (1280x640px for best display). There are also 'Template repository' and 'Repository name' fields.

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.

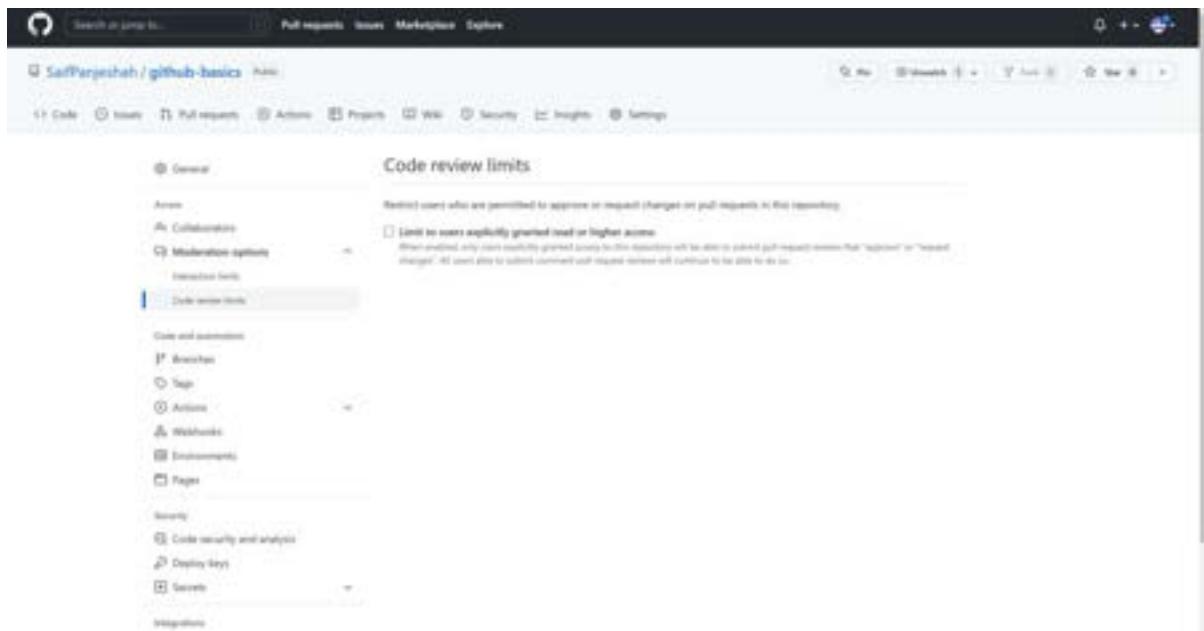


Fig. Code review limits

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

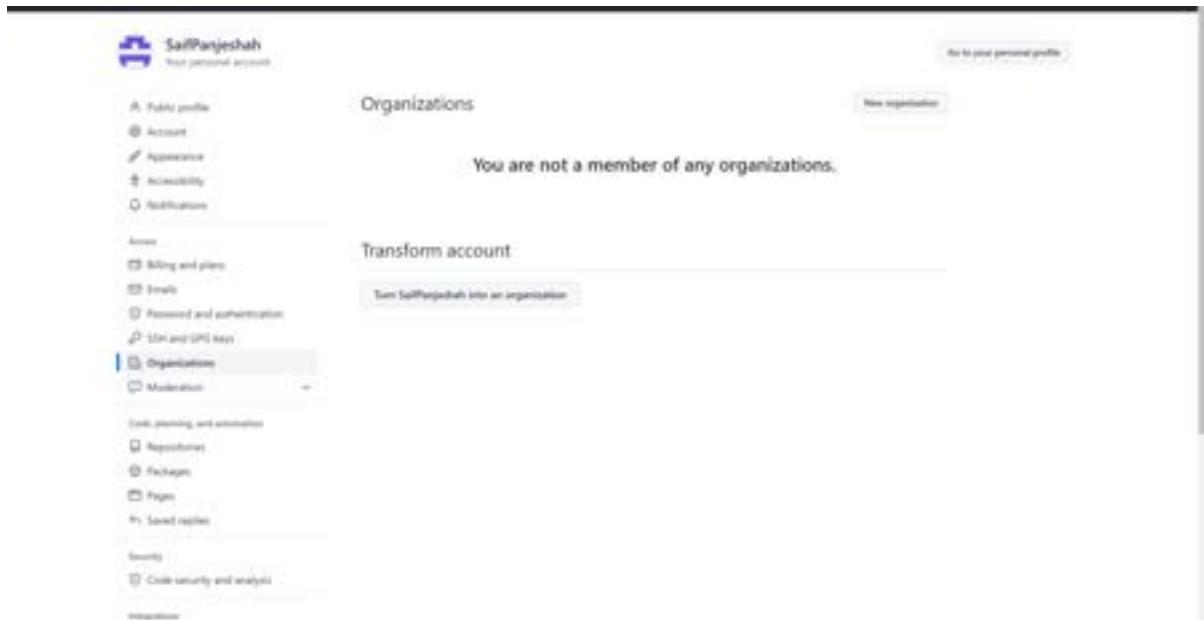


Fig. Creating organization account

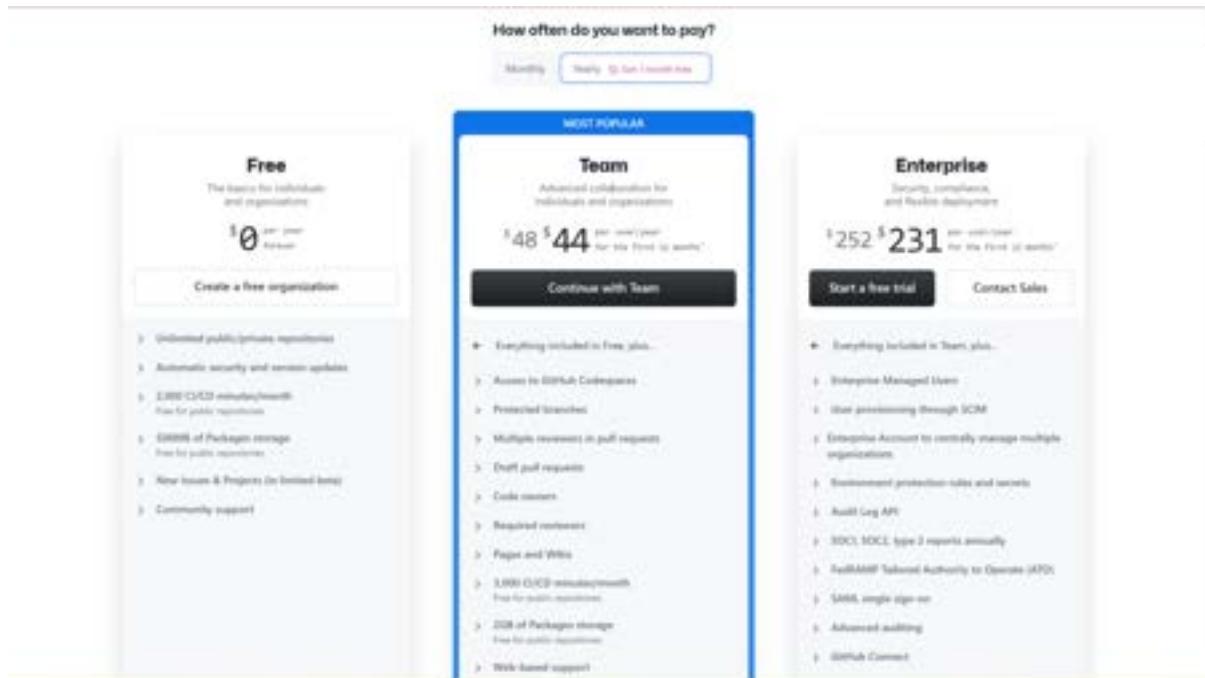


Fig. Pricing Start as Free Organization

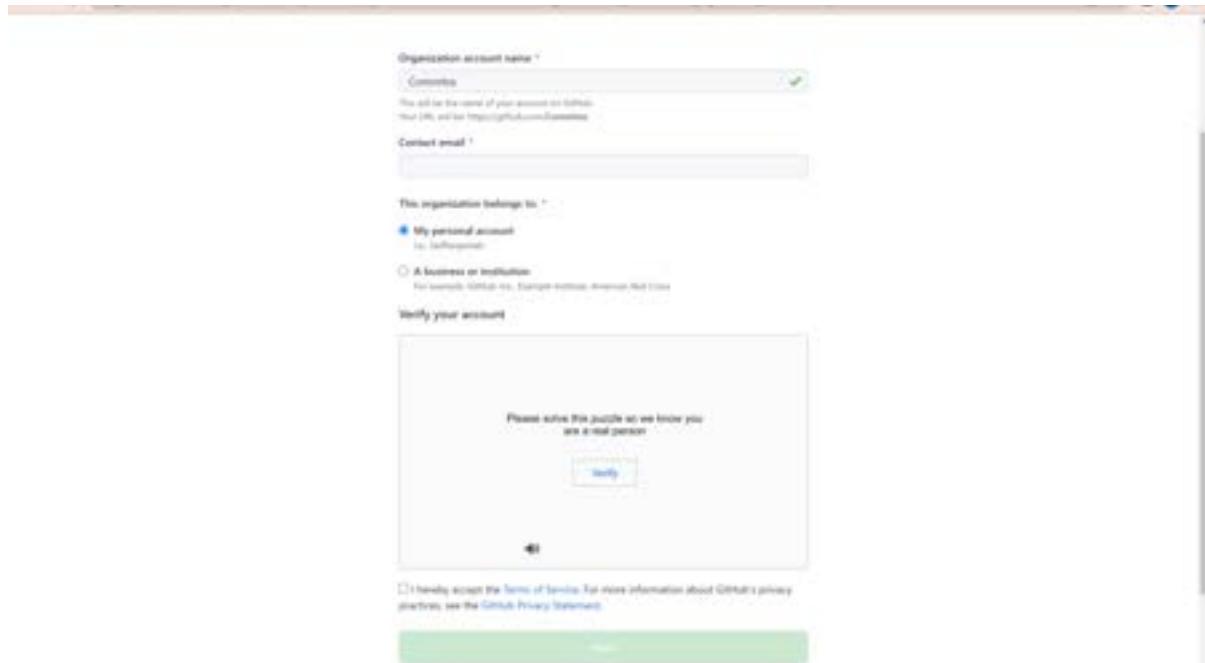


Fig. Adding Organization Name Commitsa

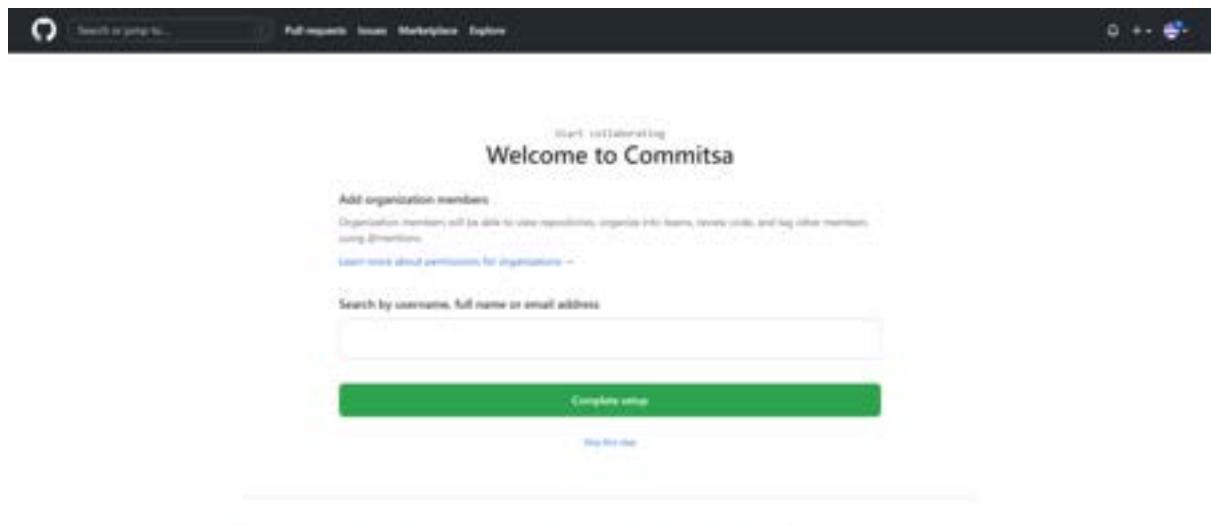


Fig. Verified Successful

A screenshot of a web browser showing the GitHub organization setup interface. The title 'Welcome to GitHub' is at the top. Below it, a message says 'Woohoo! You've joined millions of developers who are doing their best work on GitHub. Tell us what you're interested in. We'll help you get there.' There are sections for 'Tell us about you:' and 'What do you spend time on most day-to-day?'. Under 'What do you spend time on most day-to-day?', there are four options with checkboxes: 'Writing code', 'Managing and coordinating engineering work', 'Planning projects', and 'Editing administrative tasks'. Below these, there's an 'Other' section with a text input field labeled 'Please describe'. Further down, there's a section for 'Tell us about your team:' and 'How many people do you expect to actively work within this GitHub organization?'. A row of radio buttons shows options from '0' to '100+'. At the bottom, there's a section for 'What type of work do you plan to use this organization for?' with a note 'Please select at least one.' and a row of radio buttons for 'Software development', 'Data analysis', 'Machine learning', 'Cloud computing', 'Hardware engineering', 'Robotics', 'Quantum computing', 'Space exploration', and 'Bioengineering'.

Fig. Complete the Setup and submit

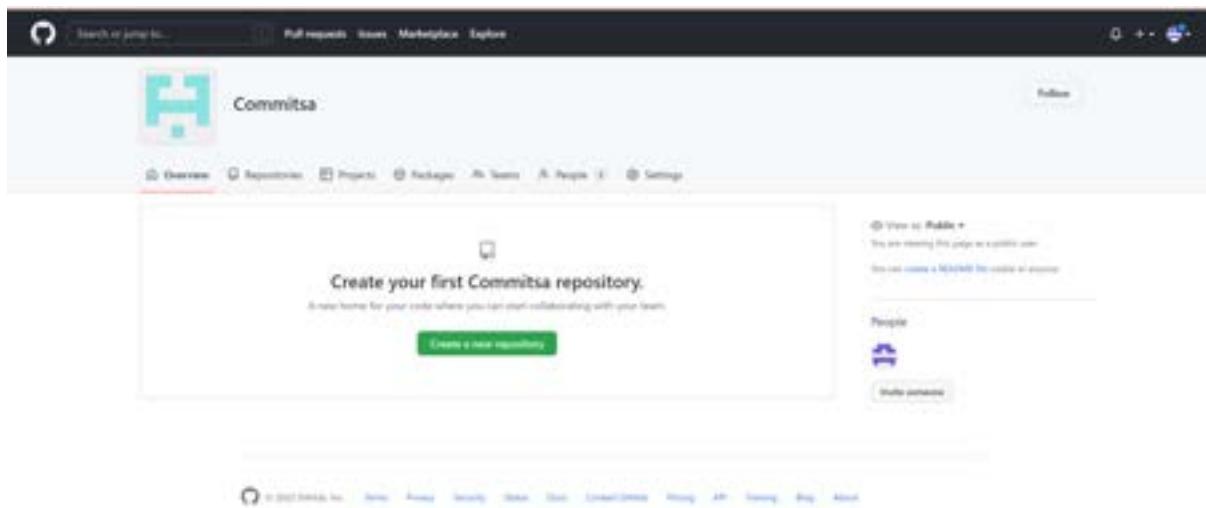


Fig. Successful Created

To View Organization from Dashboard

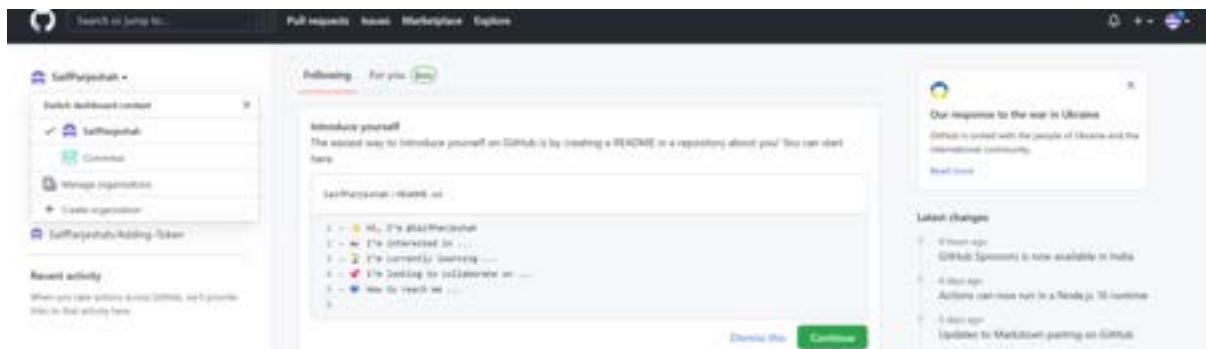


Fig. View Organization

Exploring Member Repository Permission:

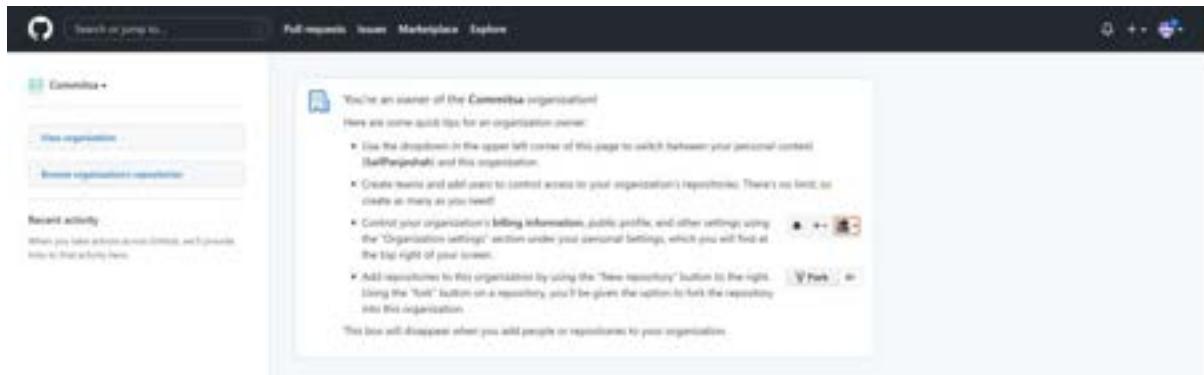


Fig. Commitsa Organization

What Missing, Here Member and repositories in Organization?

Creating New Repository:

Path: Inside Organization -> Create New Repo

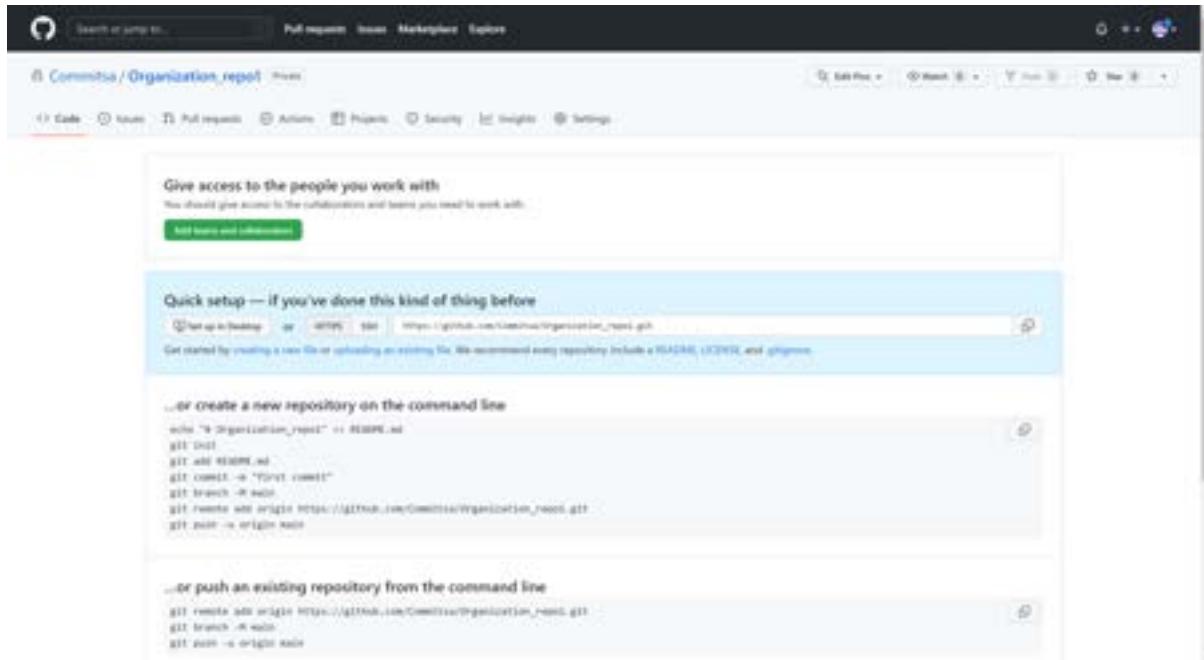


Fig. Successful Created Private Repository in Organization

After Creating repository Inviting Teams Members Same as Personal Account:

The screenshot shows the GitHub organization settings page for 'Organization_repo'. The left sidebar has sections like General, Collaboration and teams (selected), Code and automation, Security, and Integrations. The main area is titled 'Who has access' with three tabs: 'Read access' (selected), 'Write access', and 'Owner access'. Under 'Read access', it says 'Only those with access to this repository can view its history.' Under 'Write access', it says 'All 1 members can access this repository.' Under 'Owner access', it says '8 teams or members have access to this repository. Only Owners can contribute to this repository.' Below this is a 'Manage access' section with a message 'You haven't added any teams or people yet' and a 'Manage access' button.

Fig. Manage access

We can privilege our base role member (account owner)

The screenshot shows the GitHub organization settings page for 'Organization_repo'. The left sidebar has sections like General, Billing and plans, Member privileges (selected), Team discussions, Import/export, and Moderation. The main area is titled 'Member privileges' with sections for 'Base permissions', 'Repository creation', 'Repository forking', and 'Pages creation'. In 'Base permissions', there's a note about base permissions being applied to all members and external collaborators. In 'Repository creation', 'Public' is selected. In 'Repository forking', 'Allow forking of private repositories' is checked. In 'Pages creation', 'Public' is selected. A 'Save' button is 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.

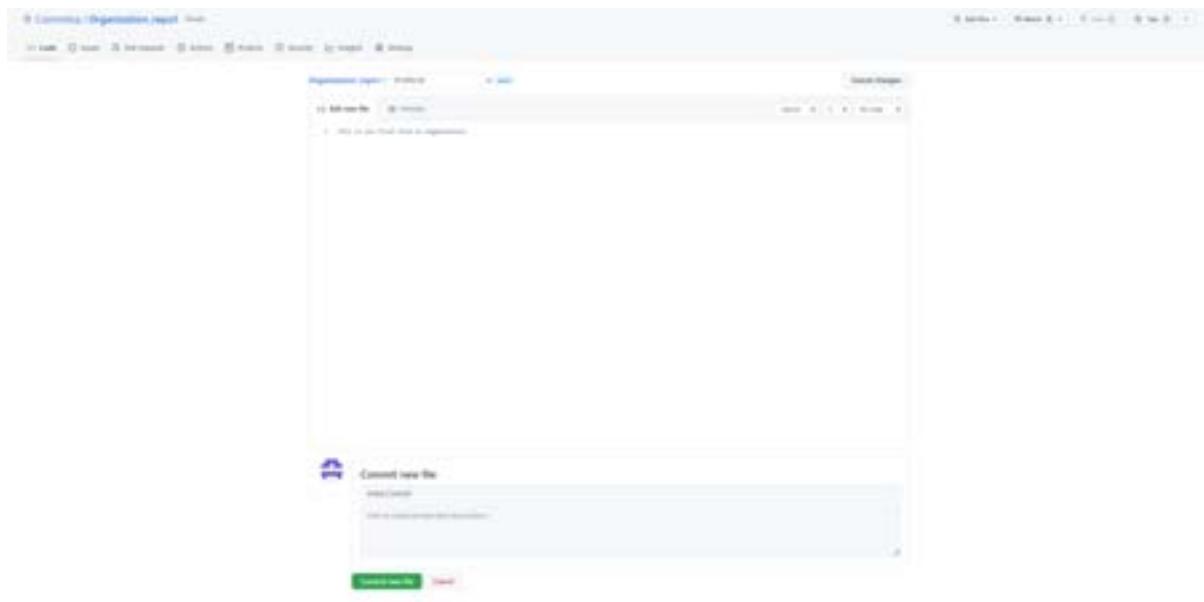


Fig. Adding First File in Organizational Account

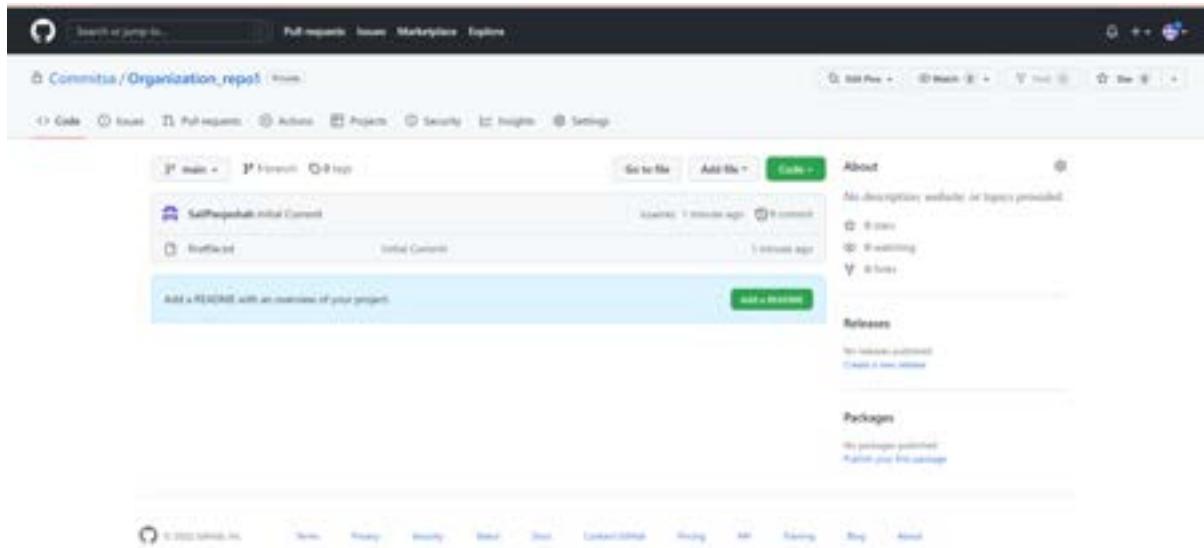


Fig. FirstFile.txt Created

The screenshot shows the GitHub organization settings page for 'Organization_repo1'. The left sidebar has sections like General, Home, Collaborators and teams (selected), Code and automation, Security, and Integrations. The main area is titled 'Who has access' and shows '1 member' with 'Read & write' access. It includes a 'Manage' button and a note about repository contributors. Below this is a 'Manage access' section with a message: 'You haven't added any teams or people yet.' It has 'Add people' and 'Add teams' buttons.

Fig. Collaborators and teams settings

This screenshot shows the 'Add people to Organization_repo1' modal window. It has a search bar and a large green 'Select 3 members' button. The background shows the same 'Who has access' interface as the previous screenshot.

Fig. Adding Outside Collaborators Organization read only permission

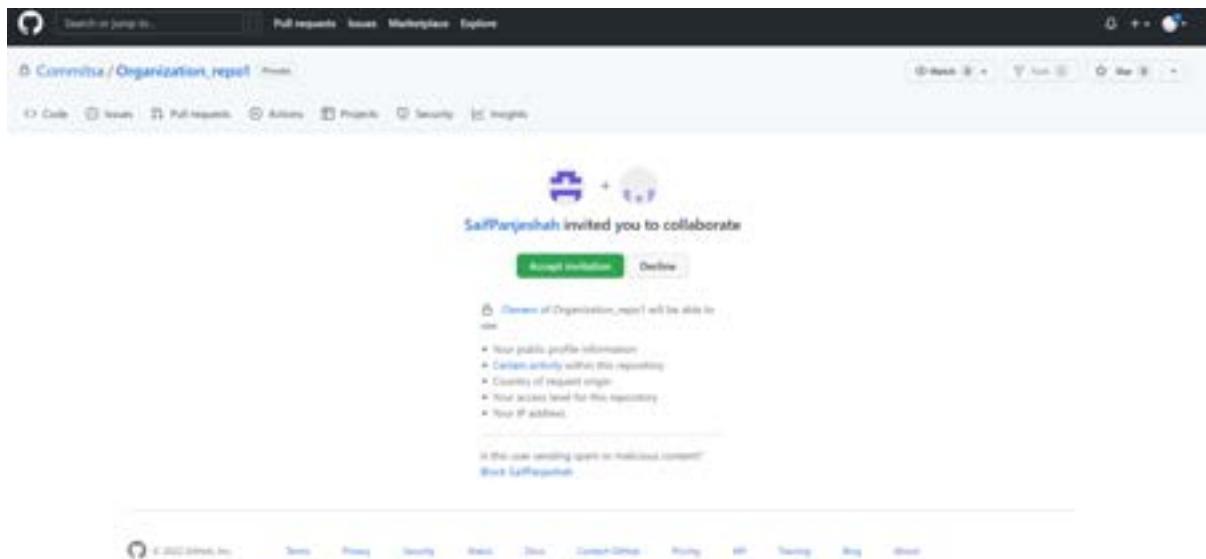


Fig. Outside Collaborators Organization Invitation

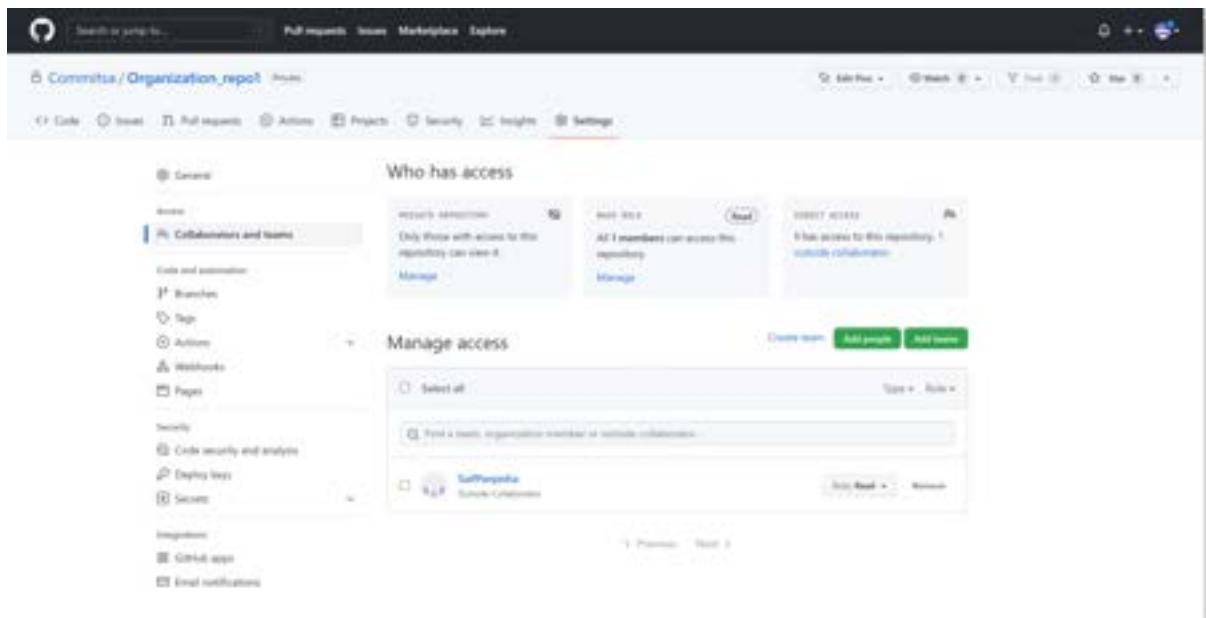


Fig. Successful added outside collaborator

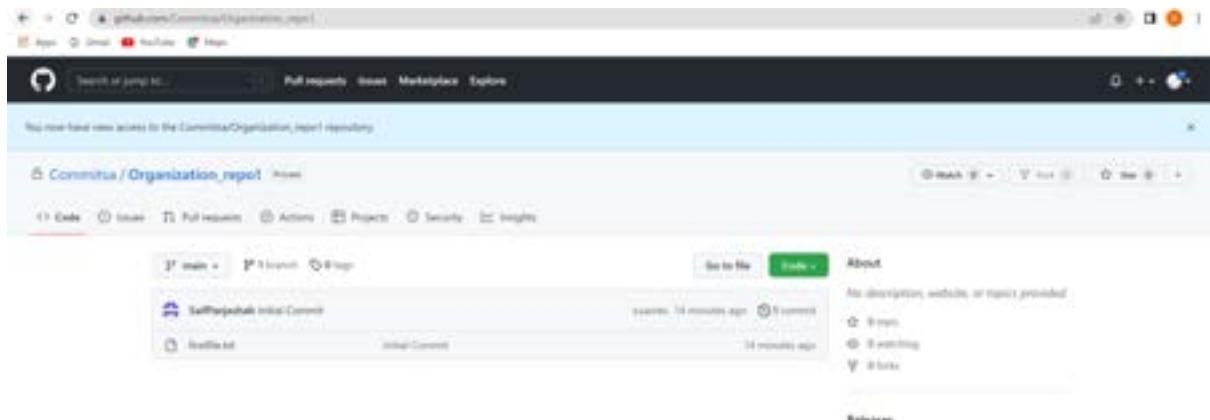


Fig. Given view access permission only

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

The screenshot shows the VS Code interface. In the Explorer sidebar, there's a folder named 'ORGANIZATION' containing two files: 'firstfile.txt' and 'secondfile.txt'. The 'secondfile.txt' file is currently selected. In the bottom right corner, there's a terminal window with the following command history:

```
D:\One Drive\data\Desktop\Organization>git clone https://github.com/Commit1a/Organization_repol.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/Commit1a/Organization_repol.git
  b2a6f82..7c93986 main -> main

D:\One Drive\data\Desktop\Organization>[]
```

Fig. Successful added to organization file

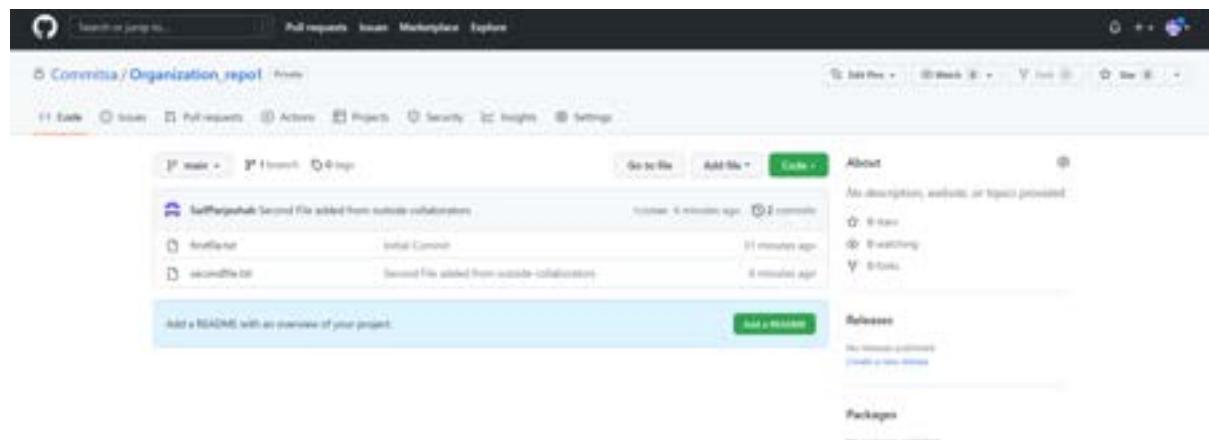


Fig. Successful added to GitHub

Adding Organization Members:

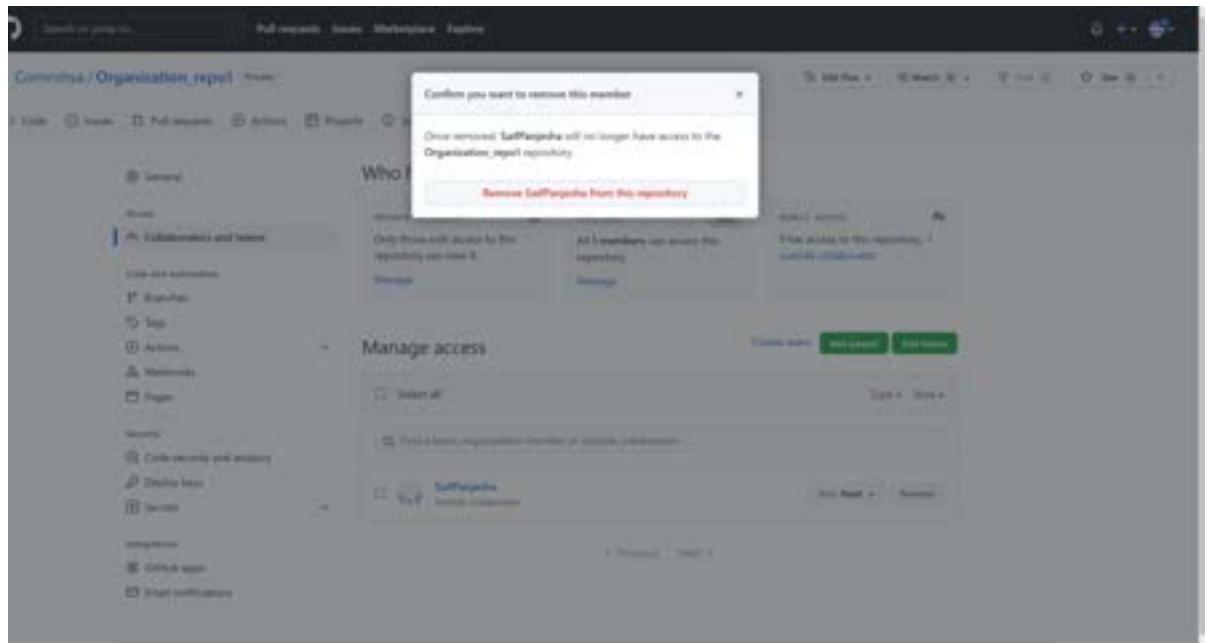


Fig. removing the outside collaborators

Let's explore people tab on Organization:

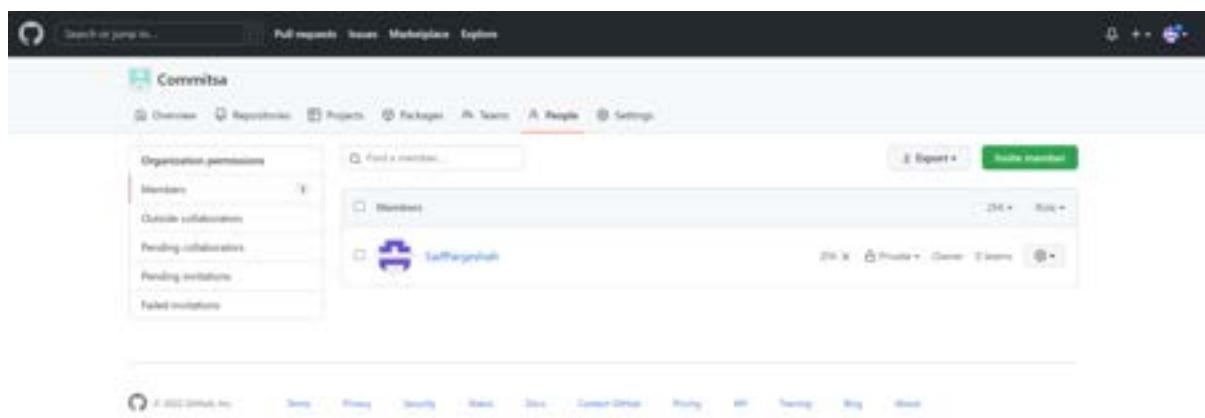


Fig. Explore people tab

Inviting Member:

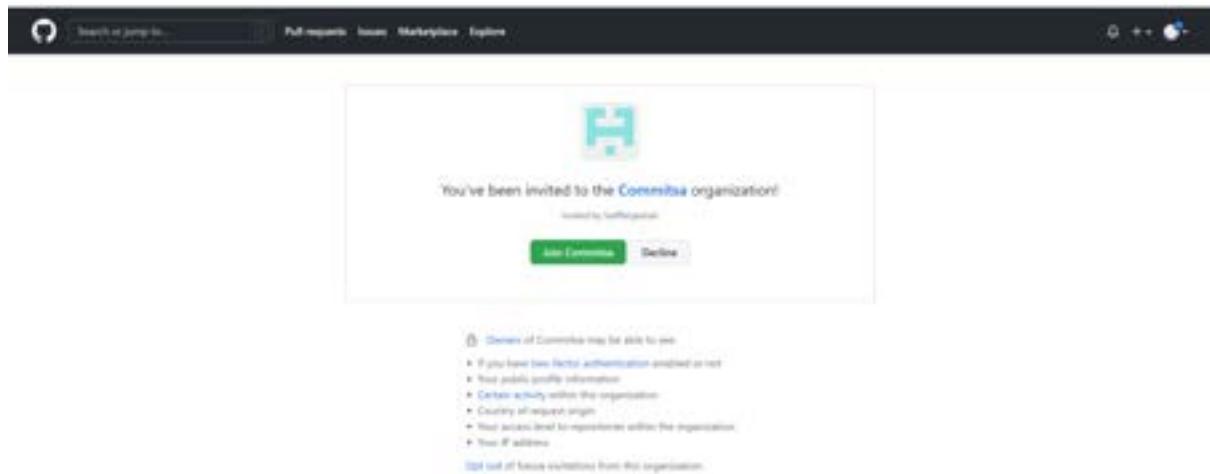


Fig. Inviting member to Commitsa organization

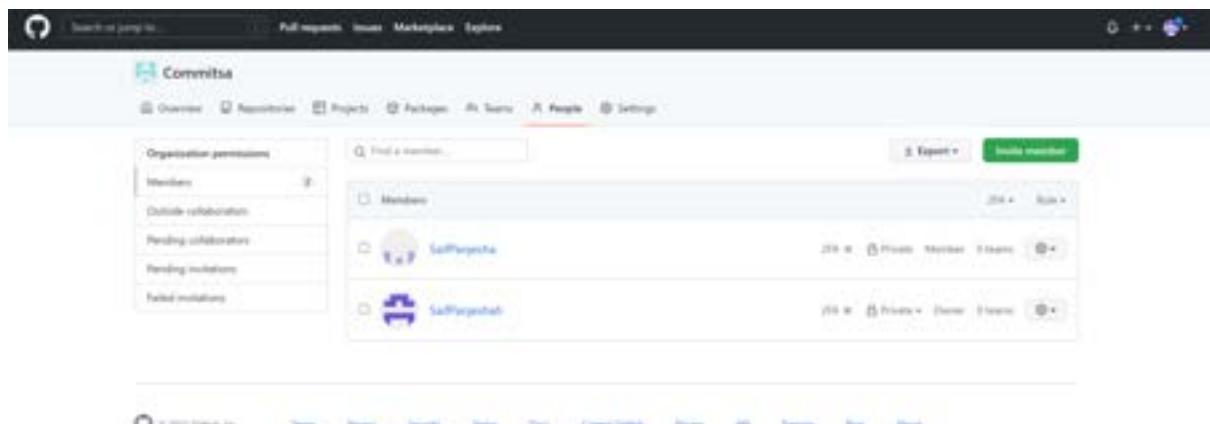
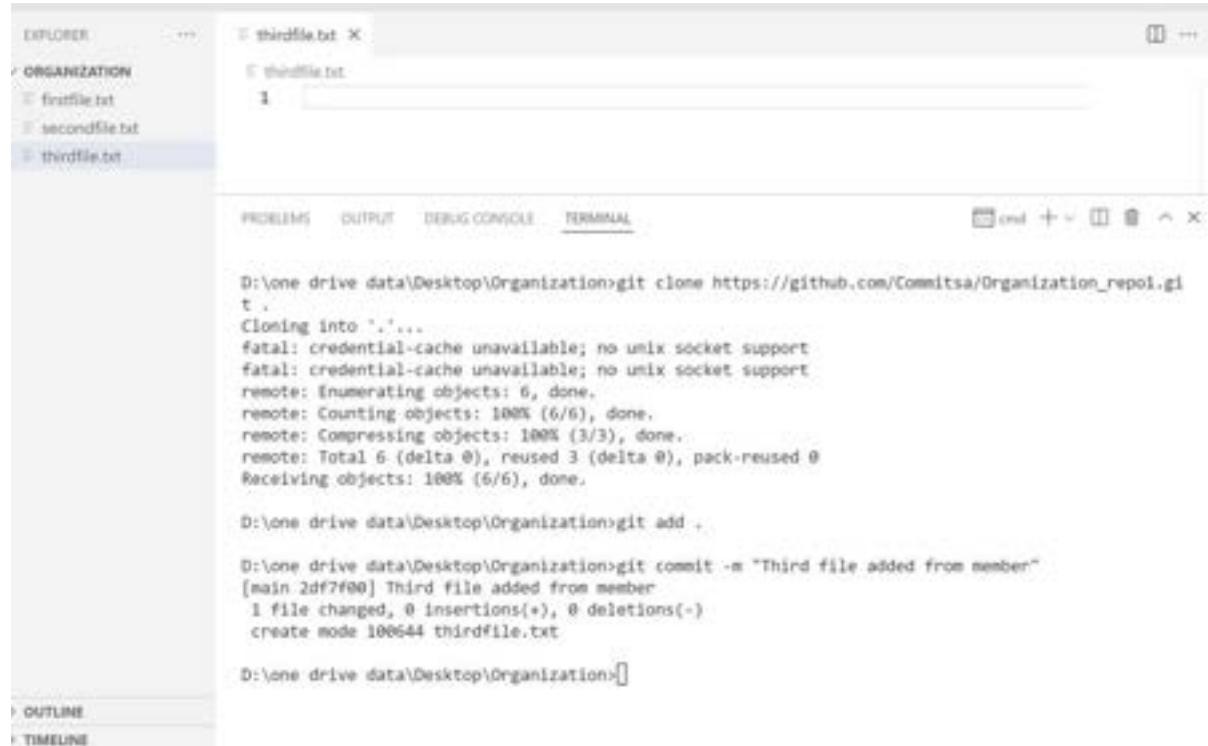


Fig. Successful Invitation

Now, trying to push the code from local branch git (VS Code) to check whether the code is successful to cloud or not.



The screenshot shows the VS Code interface with the terminal tab selected. The command history in the terminal window is as follows:

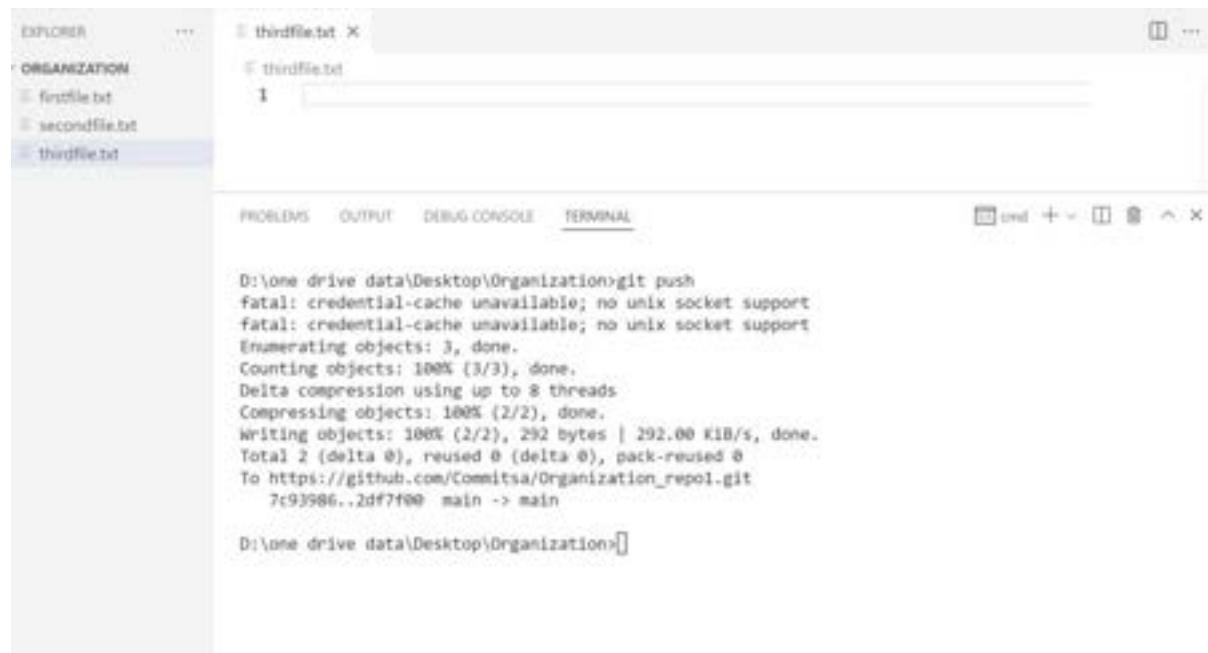
```
D:\One Drive Data\Desktop\Organization>git clone https://github.com/CommitSAs/Organization_repo1.git .
Cloning into '.'...
fatal: credential-cache unavailable; no unix socket support
fatal: credential-cache unavailable; no unix socket support
remote: Enumerating objects: 6, done.
remote: Counting 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"
[master 2df7f00] Third file added from member
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 thirdfile.txt

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

Fig. Successful clone



The screenshot shows the VS Code interface with the terminal tab selected. The command history in the terminal window is as follows:

```
D:\One Drive Data\Desktop\Organization>git push
fatal: credential-cache unavailable; no unix socket support
fatal: credential-cache unavailable; no unix socket support
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 292 bytes | 292.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/CommitSAs/Organization_repo1.git
 7c93986..2df7f00 main -> main

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

Fig. Successful Push

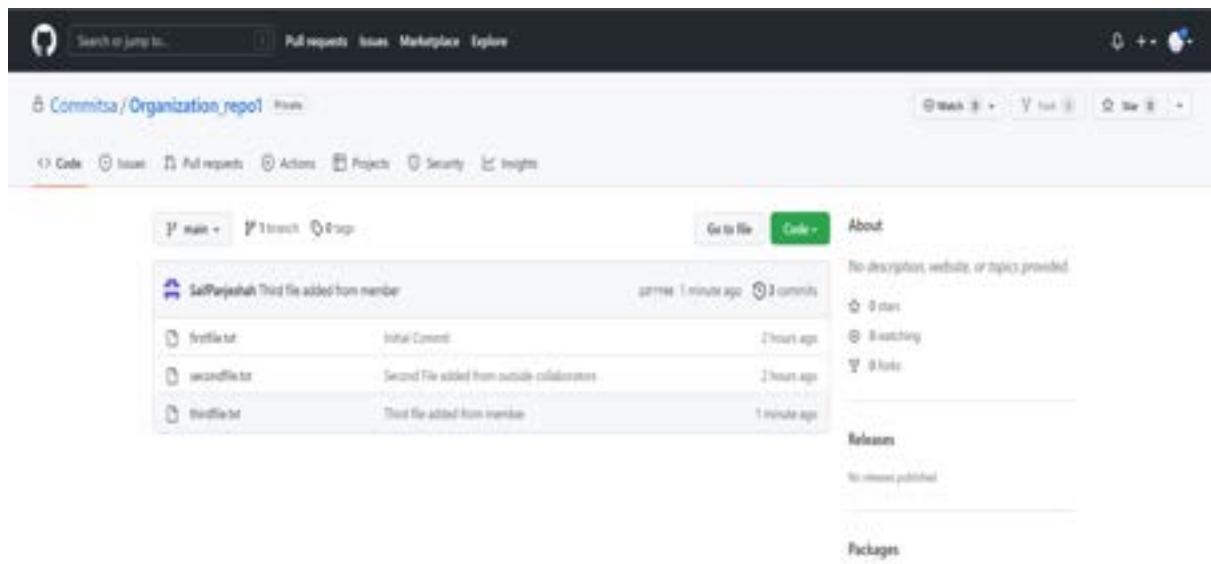


Fig. Successful Push on GitHub

Failing to Manage Access for Individual Repositories:

The screenshot shows the 'Member privileges' section of the GitHub Organization settings. On the left, a sidebar lists various organization management options like General, Billing and plans, Member privileges (which is selected), Team discussions, Import/Export, Moderation, Code planning and automation, Repository, Actions, Workflows, Discussions, Packages, Pages, Projects, Security, Authentication security, Code security and analysis, Verified and approved domains, Secrets, Integrations, and Third-party access. The main area is titled 'Member privileges' and contains sections for 'Base permissions', 'Repository creation', 'Repository forking', and 'Pages creation'. Under 'Base permissions', there are two options: 'Public' (disabled) and 'Private' (selected). Under 'Repository creation', there are also two options: 'Public' (disabled) and 'Private' (selected). Under 'Repository forking', there is one option 'Allow forking of private repositories' (disabled). Under 'Pages creation', there is one option 'Public' (selected).

Fig. Giving Member privileges access to Write Option

Creating New Repositories in Organizational Account for read only members:

The screenshot shows the GitHub repository creation interface. At the top, there are tabs for 'Code', 'Issue', 'Pull request', 'Issues', 'Merge', 'Merge', 'Issues', 'Merge', and 'Settings'. Below the tabs, it says 'Create new repository'. A dropdown menu shows 'Organization: Commitus' and 'Member: member'. The main area has fields for 'Repository name' (set to 'member') and 'Visibility' (set to 'Public'). Below these, there's a 'Commit new file' section with a 'Commit message' field containing 'Hello world!' and a 'Commit description' field containing 'Initial commit'. At the bottom, there are two buttons: 'Commit now' (green) and 'Cancel'.

Fig. Creating member read only repositories

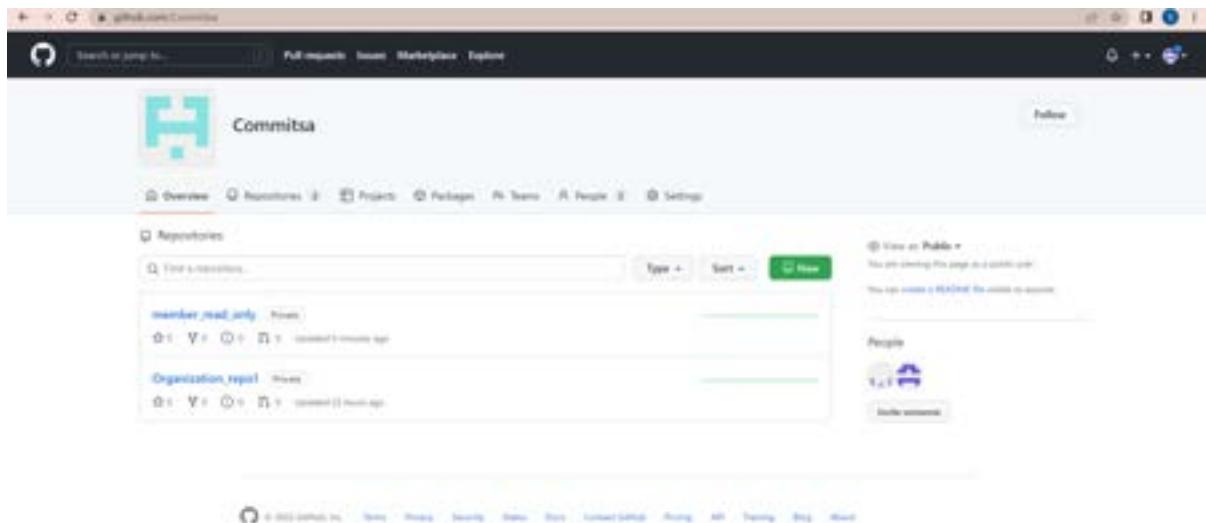


Fig. Successful Created Account

Checking Access for Individual Repositories:

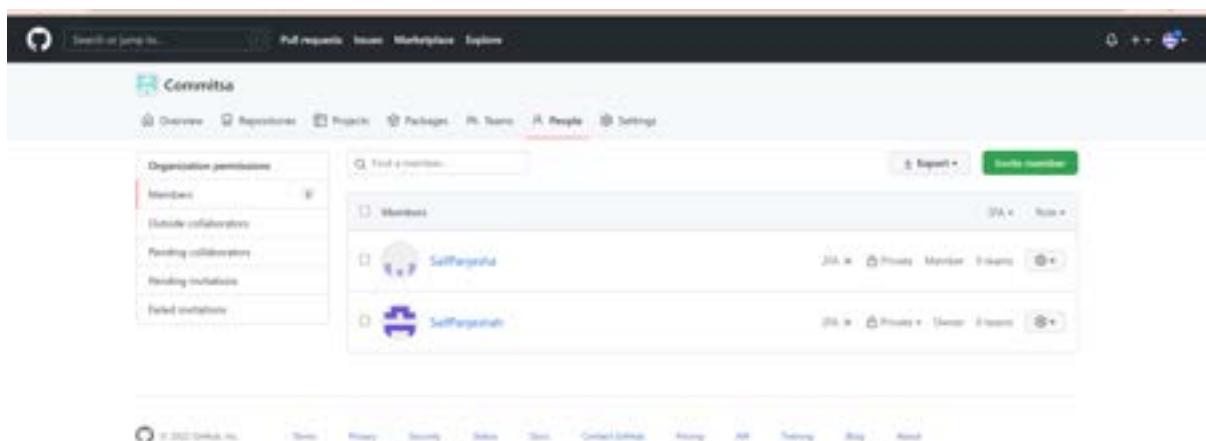


Fig. Checking access

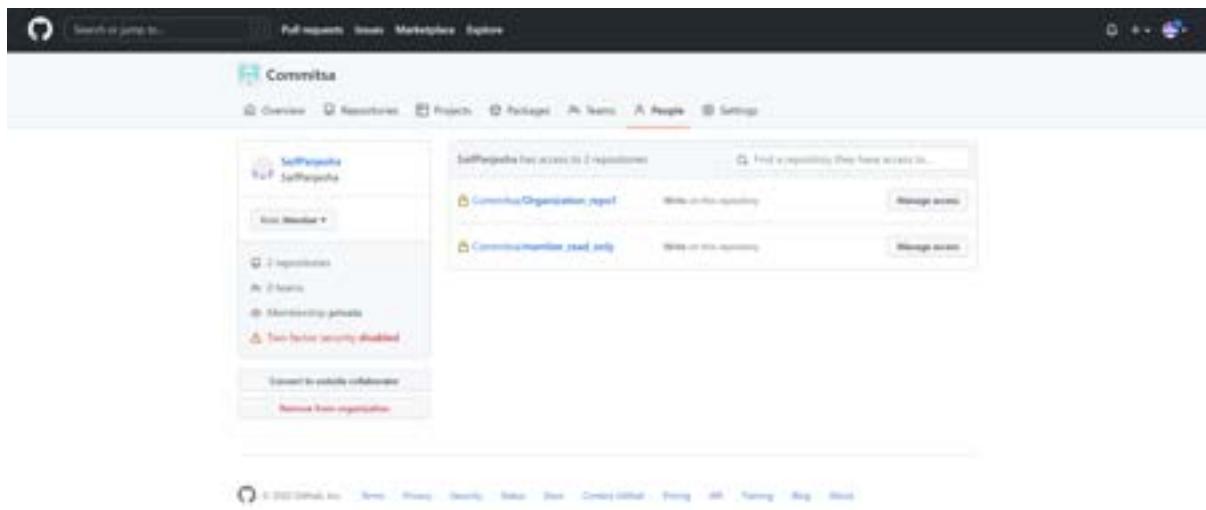


Fig. Shows both the account has same access for repositories

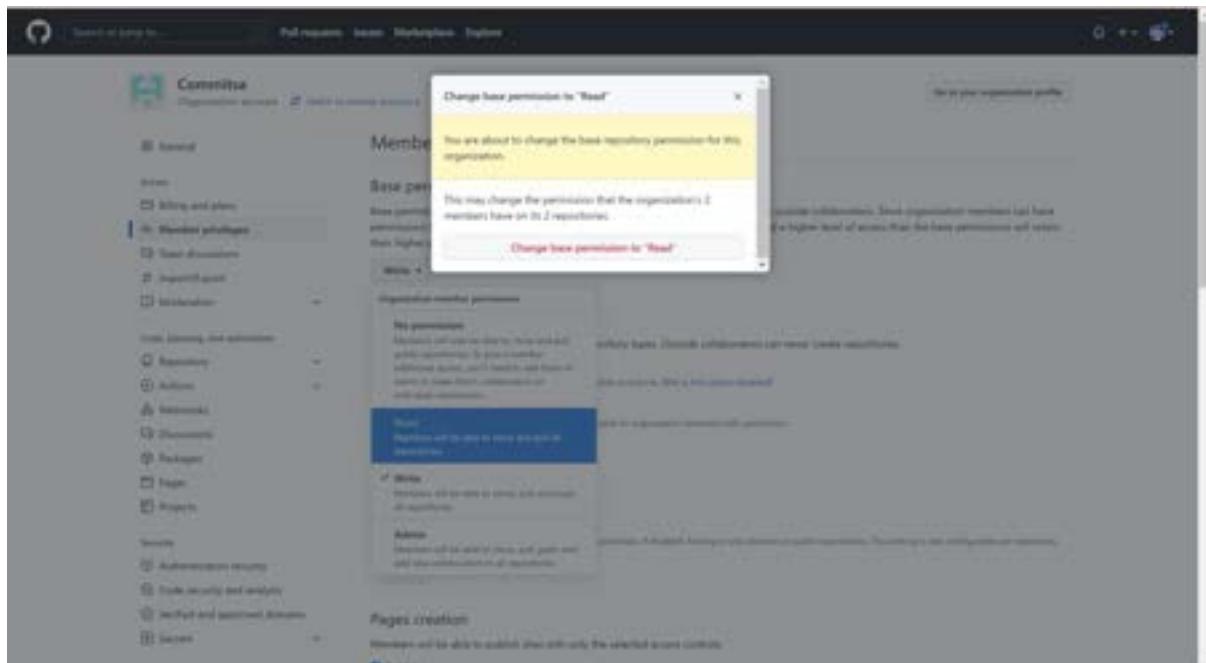


Fig. Shows Permission may change for both the repositories

How to resolve this problem access Permission?

By Introducing Teams in Organizational Account:

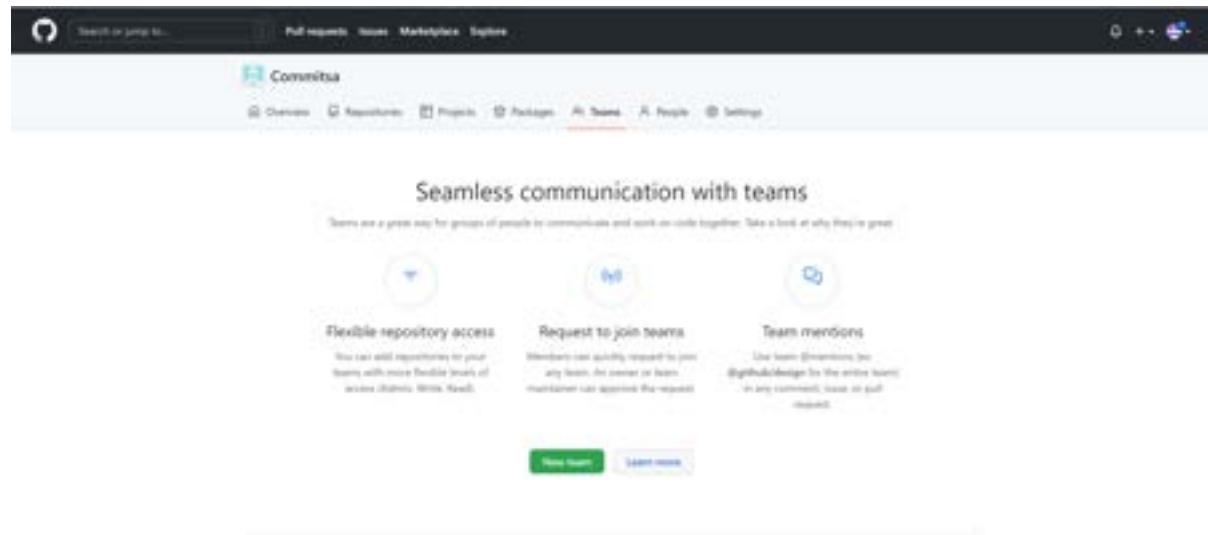


Fig. Teams Ribbons of Organizational Account

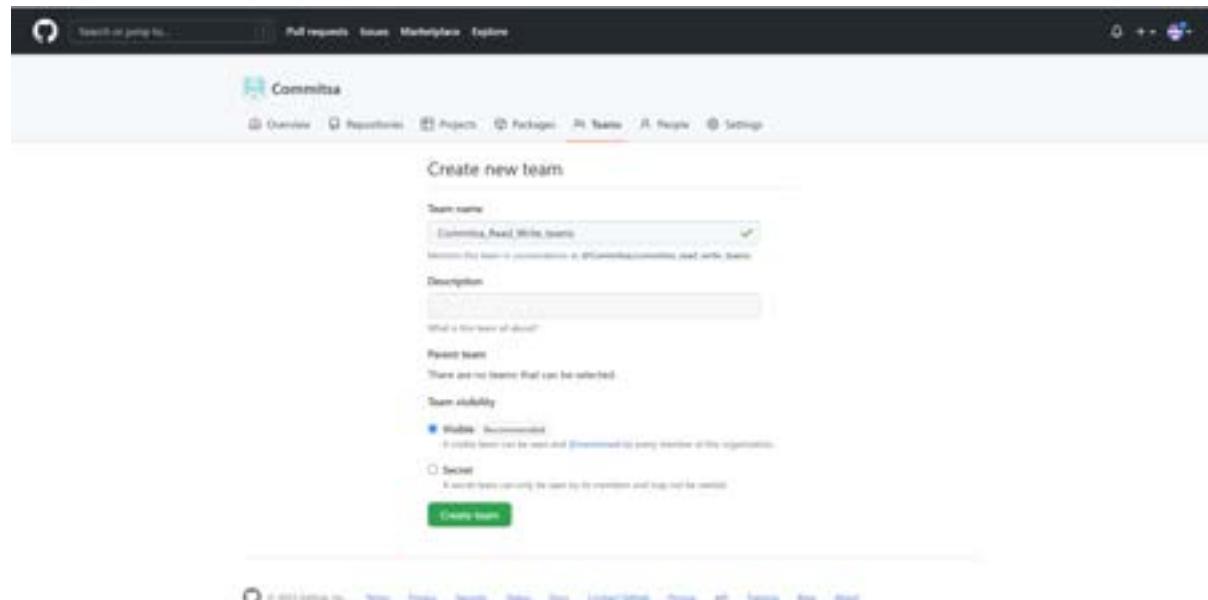


Fig. Creating Teams in Organization

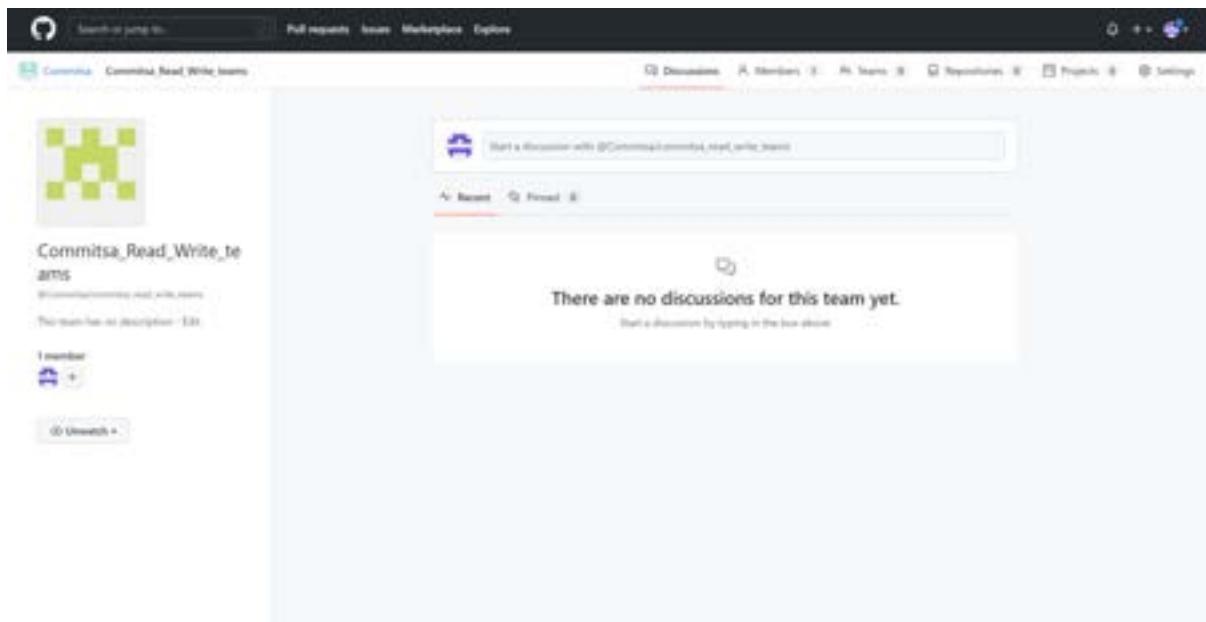


Fig. Read and Write team successfully Created

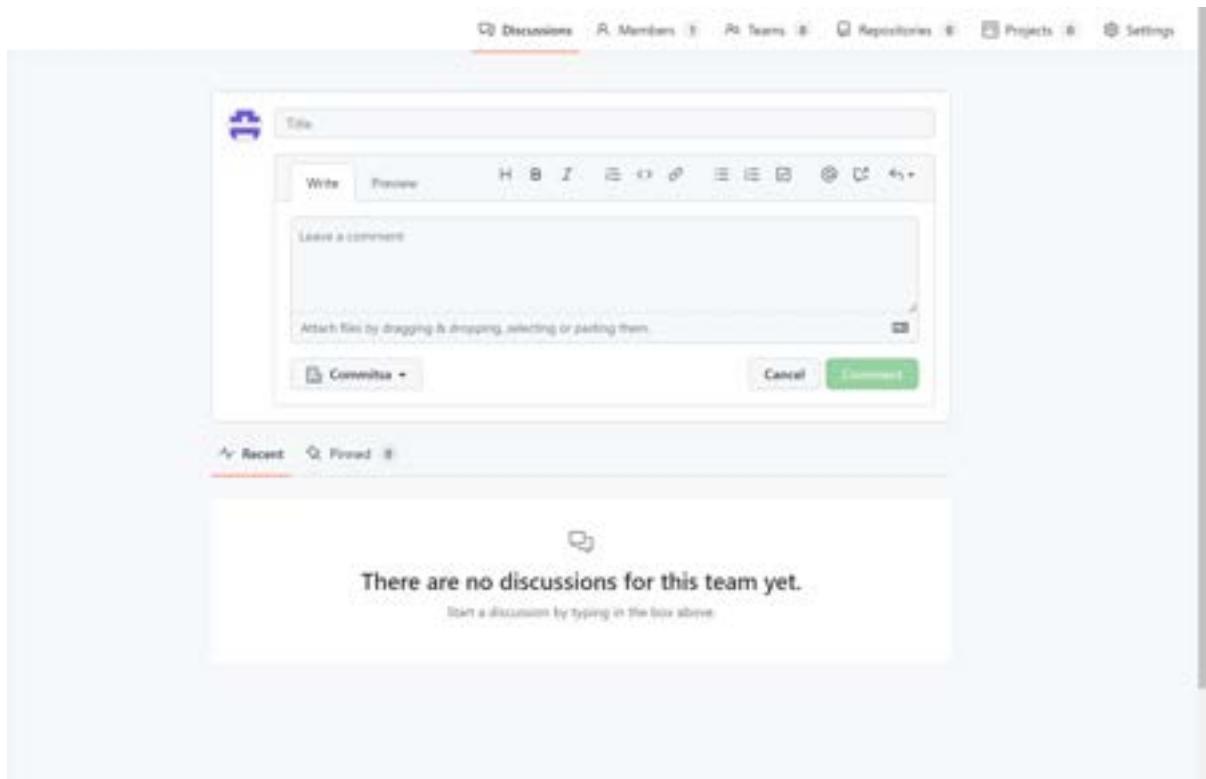


Fig. Forum of discussion ribbons in teams to check

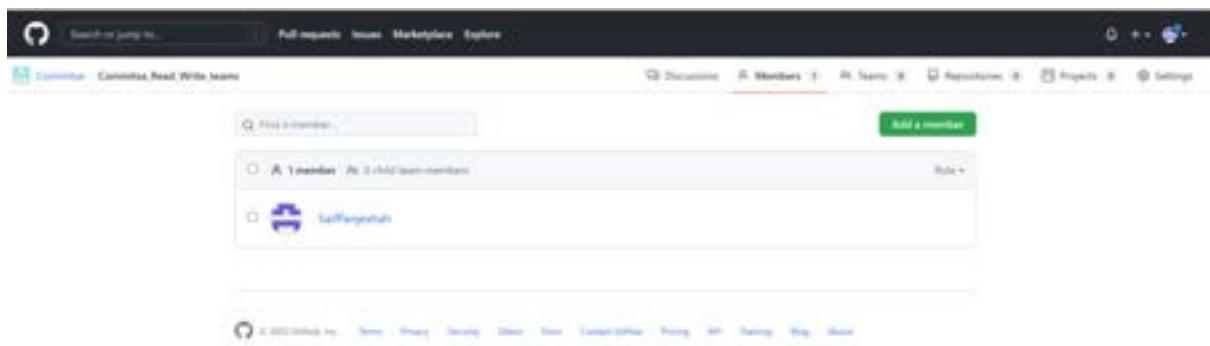


Fig. Owner as Team Member Default

Let's create team repository:

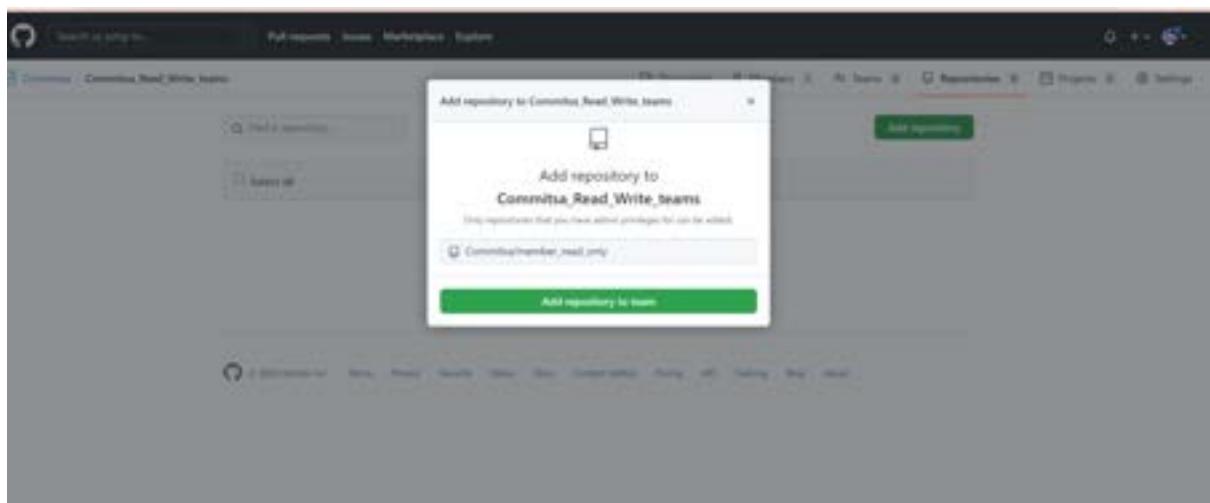


Fig. Creating Team Repository

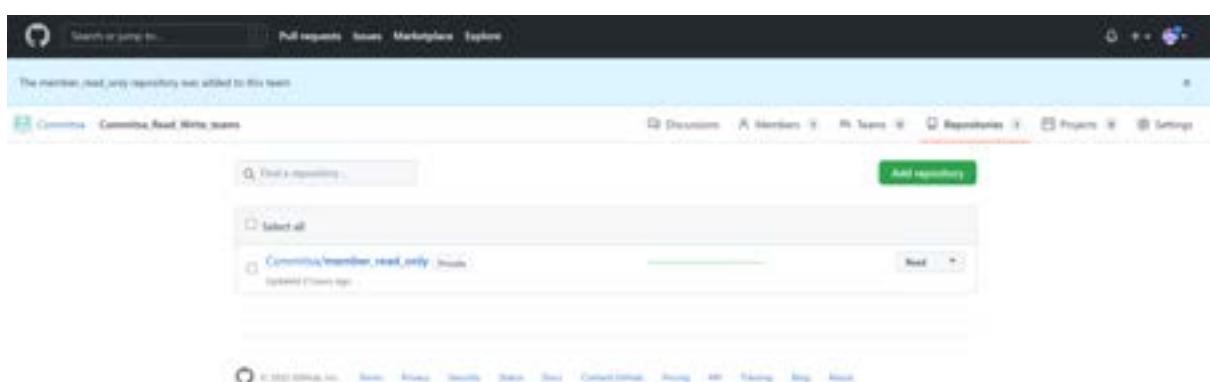


Fig. Repository Created with read access

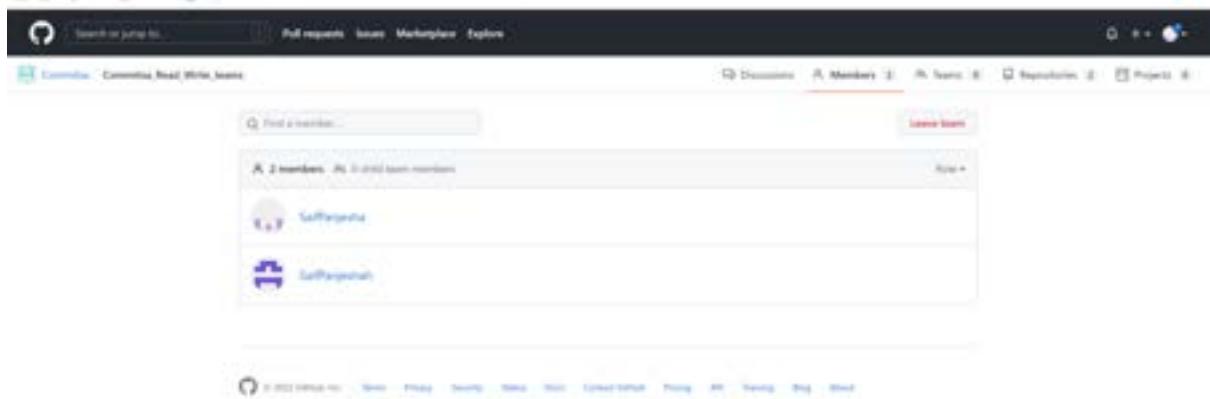


Fig. Successful adding team member

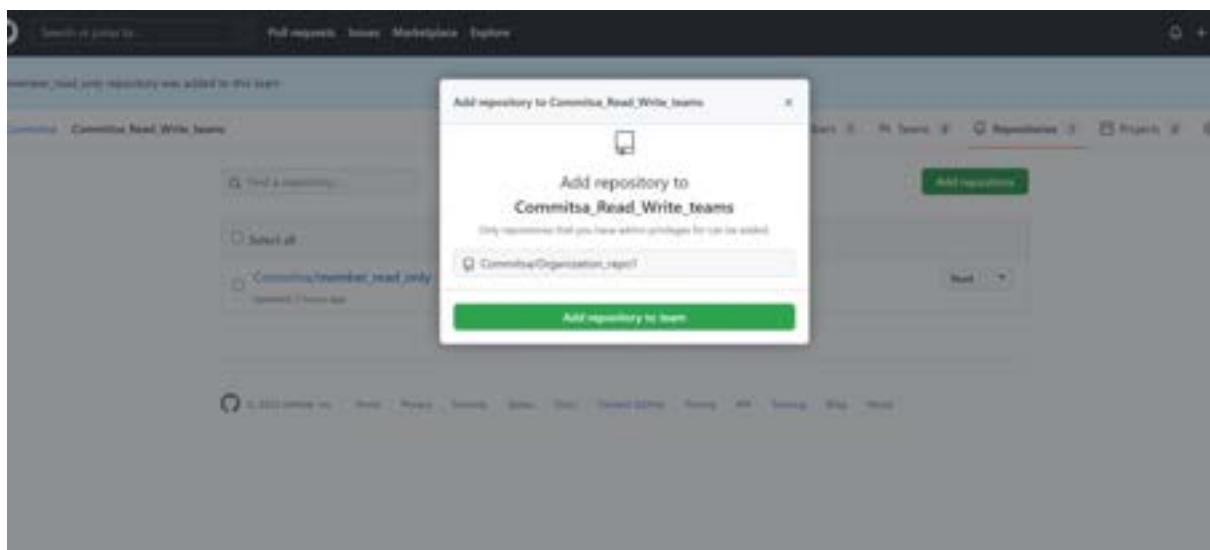


Fig. Creating Team Repository

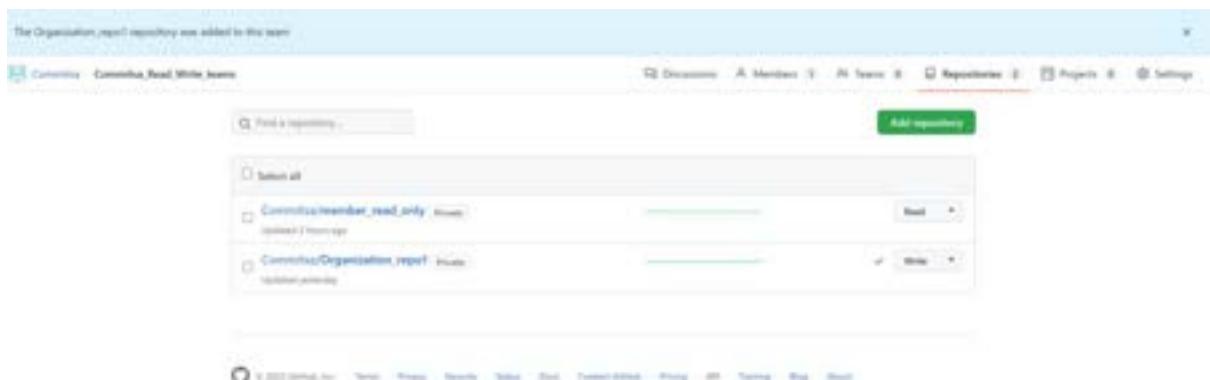
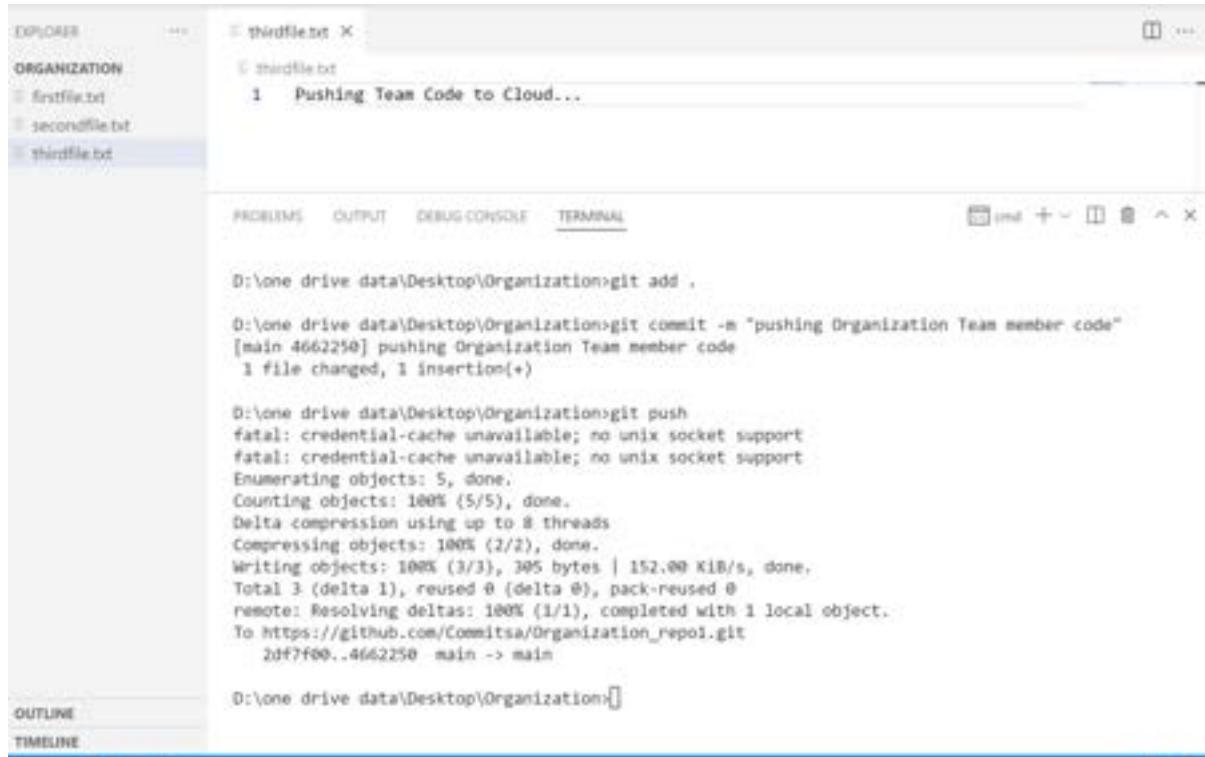


Fig. Successful Created repositories with write access

Hence, we can able to access for Individual Repositories with teams as flexible work.

Managing Team Repositories Access Efficiently:

Let's Now, trying to push the code from local branch git (VS Code) to check whether the code is successful to cloud or not.



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

```
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), 395 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  
 2d7f00..4662250 main -> main  
  
D:\One Drive data\Desktop\Organization>
```

Fig. Successful Push coding on Organization repository with write access

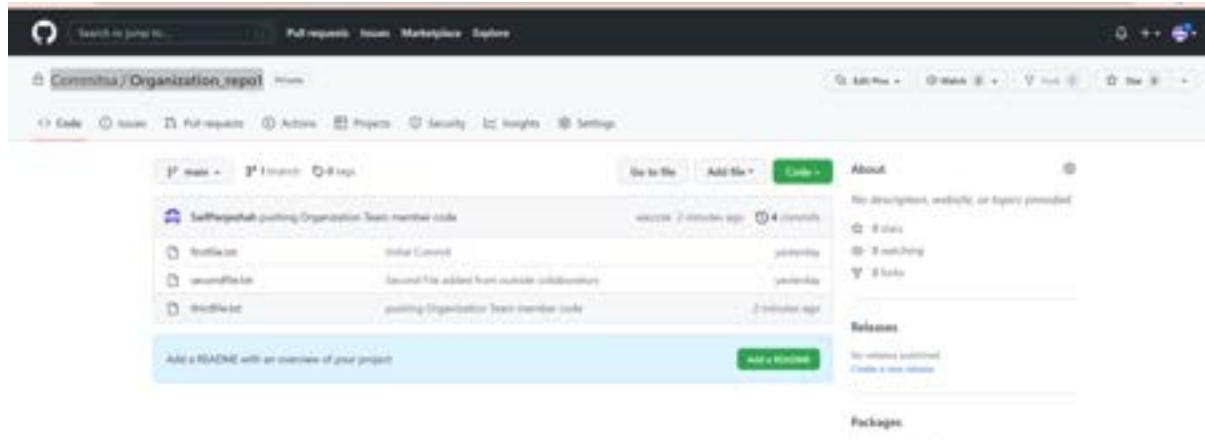


Fig. Successful Push on GitHub

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



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

- Explorer:** Shows a file named "read.txt" containing the text:

```
1 This is the First Text!
2 Let's Push Code on Cloud !!!
```
- Terminal:** Displays the command-line output of the git push process:

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

D:\One Drive\data\Desktop\Organizations>git clone https://github.com/Commitssa/member_read_only.git .
Cloning into '.'...
fatal: credential-cache unavailable; no unix socket support
fatal: credential-cache unavailable; no unix socket support
remote: Enumerating objects: 3, done.
remote: Counting objects: 1000 (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: 1000 (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 314 bytes | 314.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Commitssa/member_read_only.git
 45d1833..696f26a main -> main
```
- Outline and Timeline:** Standard VS Code navigation panes.

Fig. Successful Push coding on member_read_only repository with read access

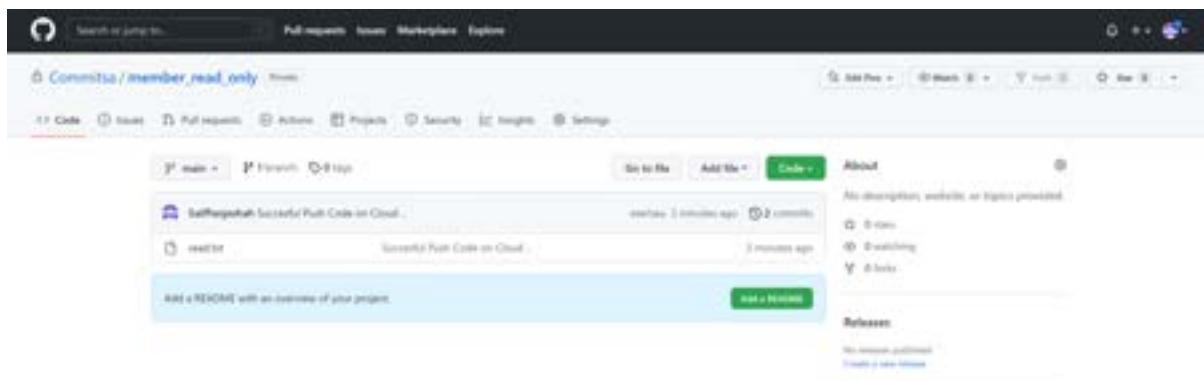


Fig. Successful Push on GitHub

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 ' <code>git clone</code> ' and it is a process of receiving all the code files to the local machine

Fig. Difference between Fork & Clone

When to use: A fork is a rough copy of a repository. Forking a repository allows you to freely test and debug with changes without affecting the original project. One of the excessive uses of forking is to propose changes for bug fixing. To resolve an issue for a bug that you found, you can:

- Fork the repository.
- Make the fix.
- Forward a pull request to the project owner

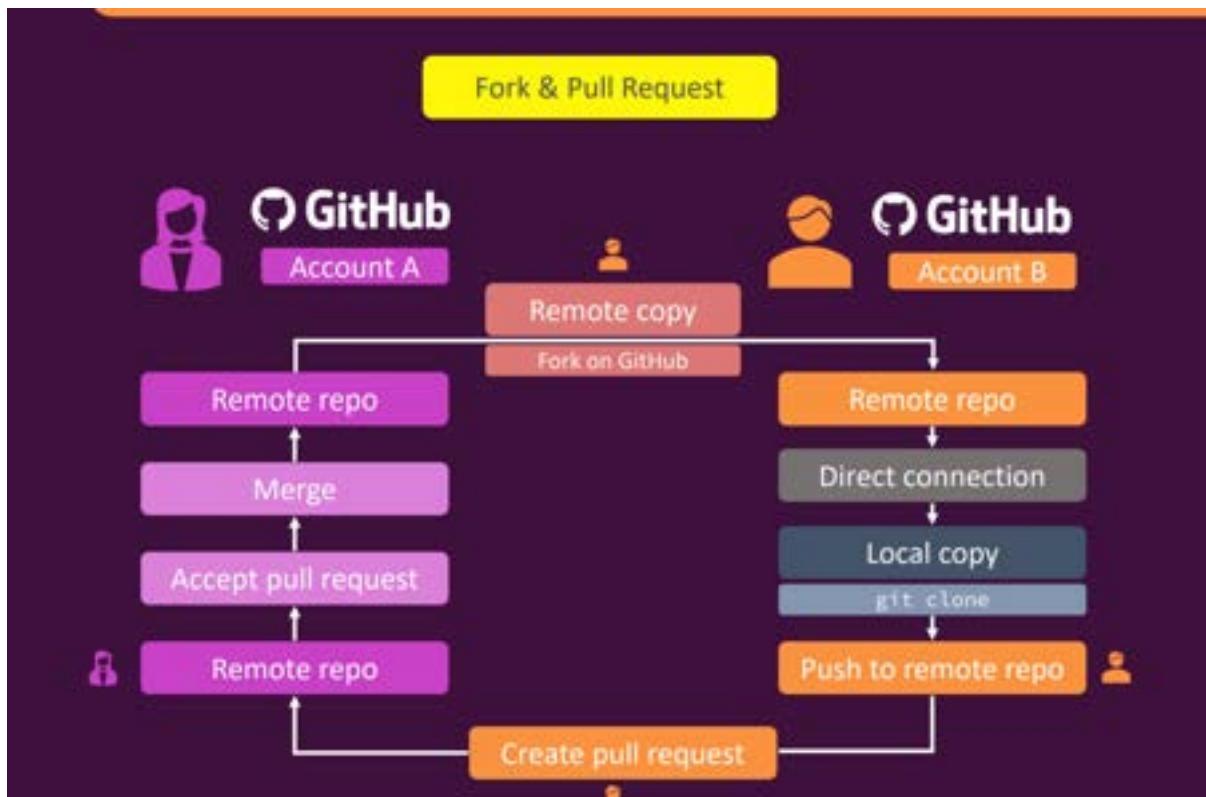


Fig. Creating Fork & Pull Request

1. Forking the Repository:

Assuming you're using GitHub, this step is easy. Just find the repository you're contributing to and press the Fork button in the upper right. This will create an exact copy of the repository (and all of its branches) under your own username

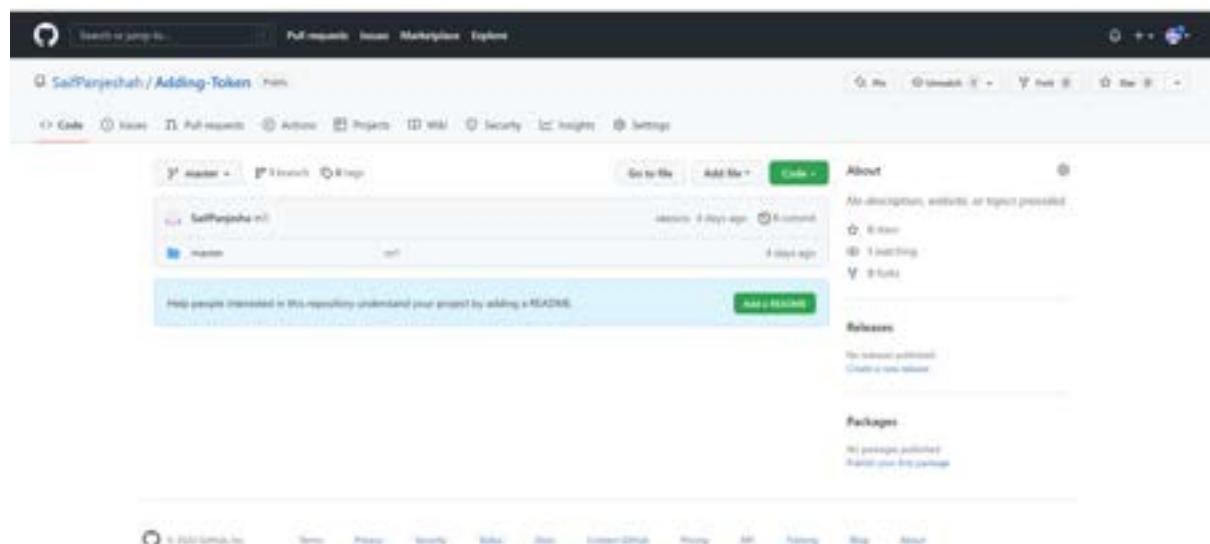


Fig. Forking the repository from another account

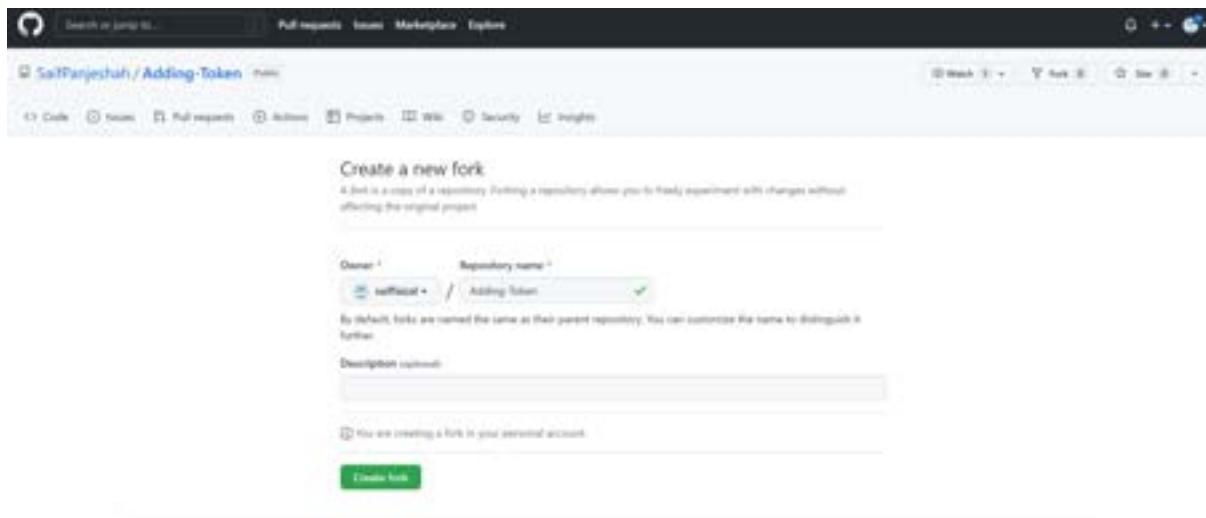


Fig. Create a new fork

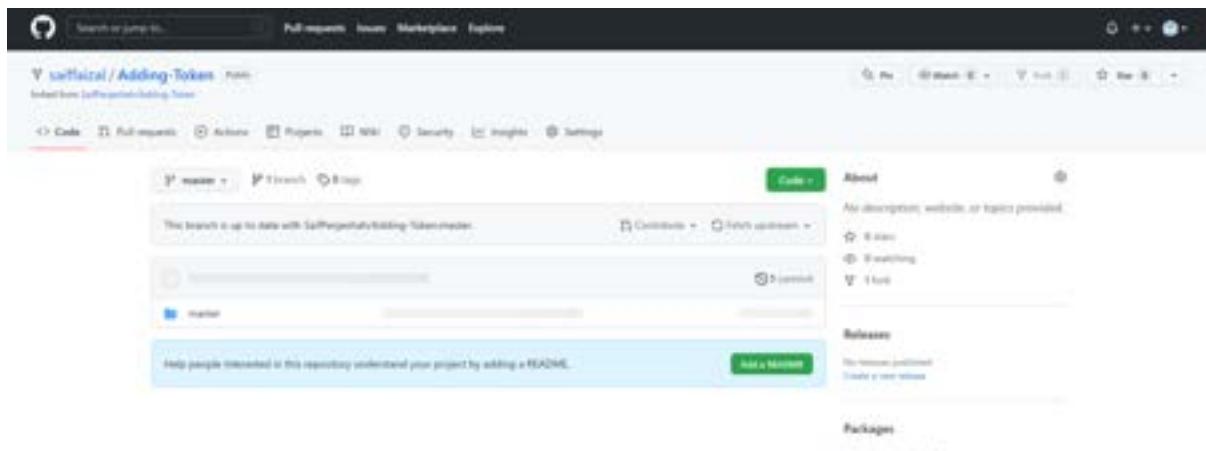


Fig. Forked Successful

2. Clone your new fork locally

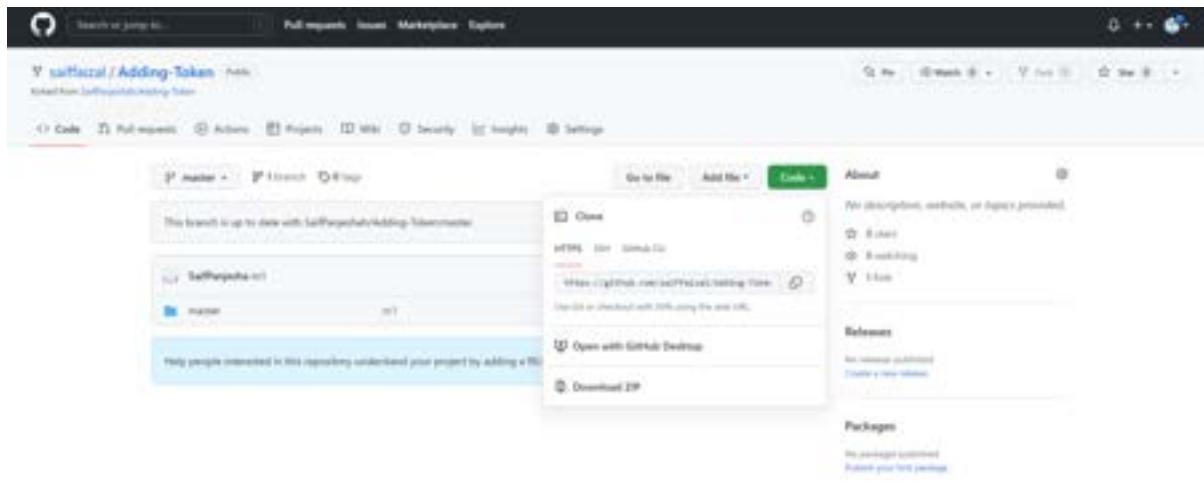


Fig. cloning the repository.

Let's Now, trying to push the code from local branch VS code console to check whether the code is successful to cloud or not.

```
D:\One Drive\data\Desktop\Forking Repositories>git clone https://github.com/saifFazal/Adding-Token.git
Cloning into '.'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100%, 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), 338 bytes | 338.00 KiB/s, done.
Total 4 (delta 0), reused 1 (delta 0), pack-reused 0
To https://github.com/saifFazal/Adding-Token.git
    4882631..7444ed8  master -> master
```

Fig. Success to push the code

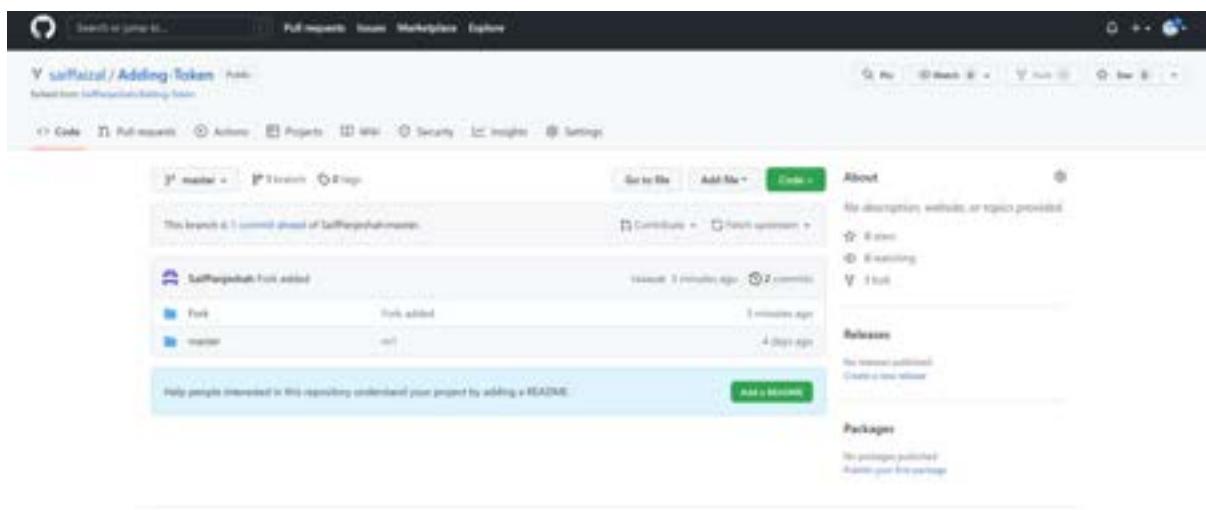


Fig. Successful on GitHub to member account

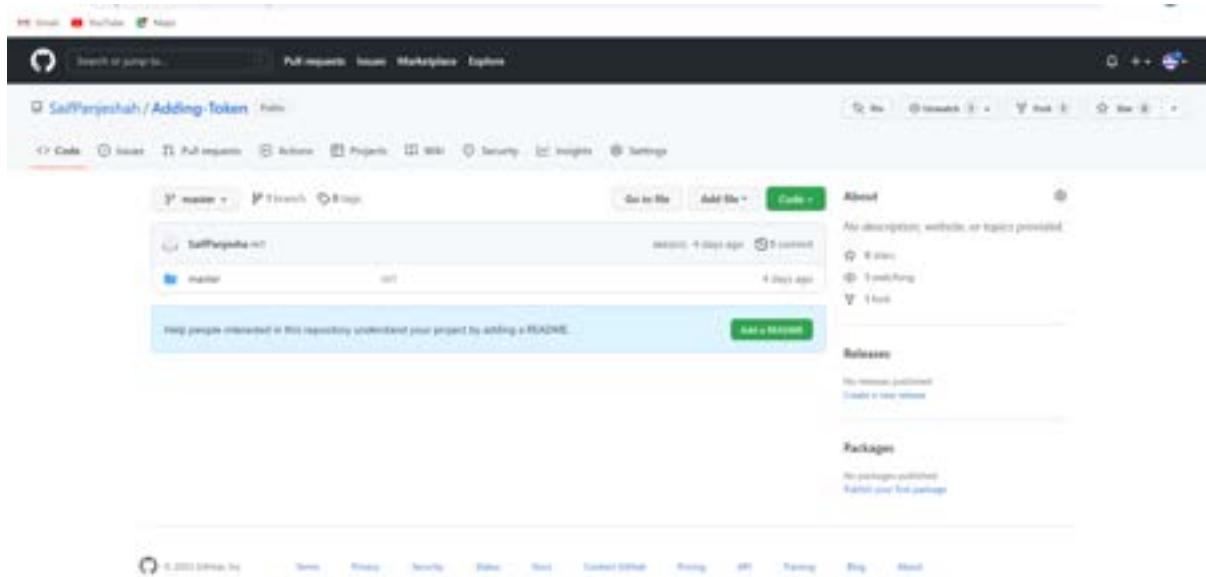


Fig. Not Fork Update Successful on GitHub owner account

How To resolve this problem?

By using pull requests in practise:

Creating a pull request

1. Switch to the branch that you want to create a pull request for [Saiffaizal/Adding-Token](#)
2. Click **Create Pull Request**. GitHub Desktop will open your default browser to take you to GitHub

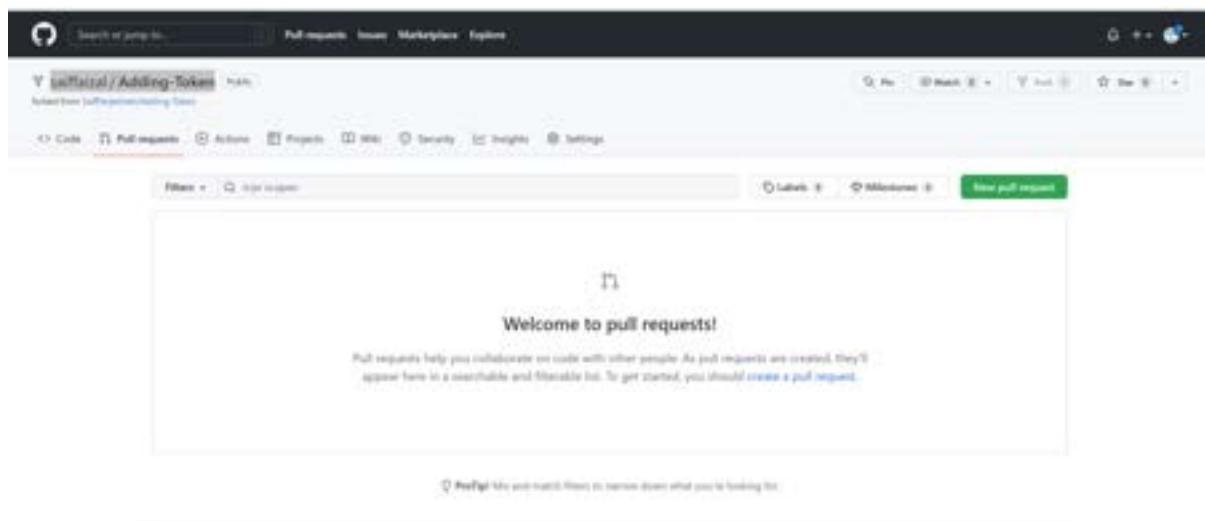


Fig. Creating a pull request

3. On GitHub, confirm that the branch in the **base:** drop-down menu is the branch where you want to merge your changes. Confirm that the branch in the **compare:** drop-down menu is the topic branch where you made your changes.

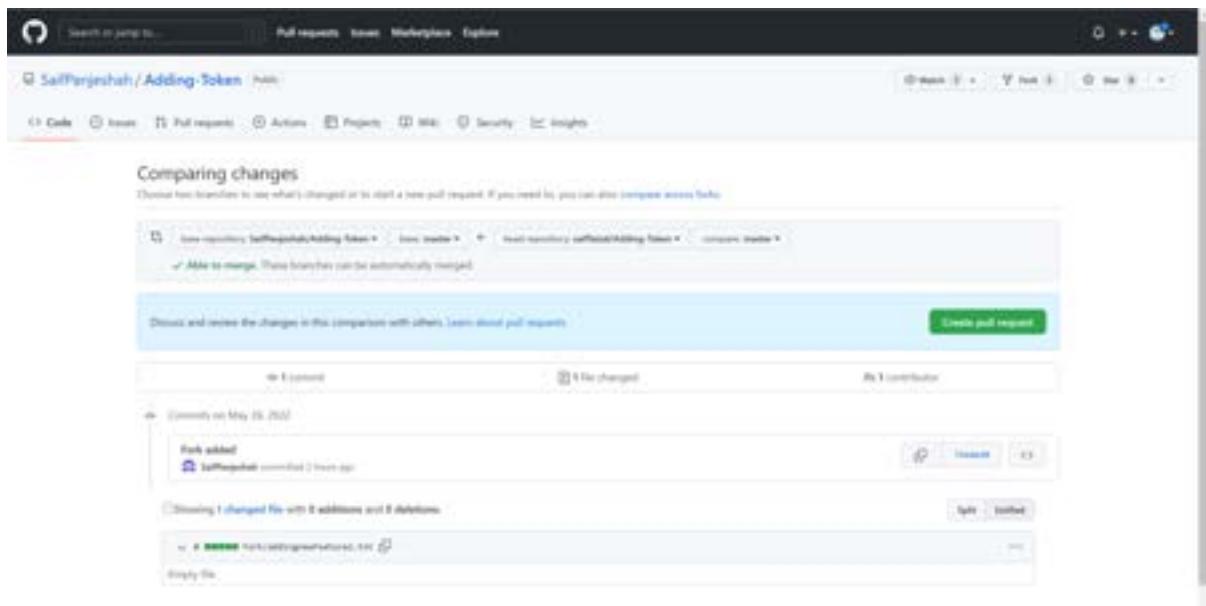


Fig. Comparing Changes

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

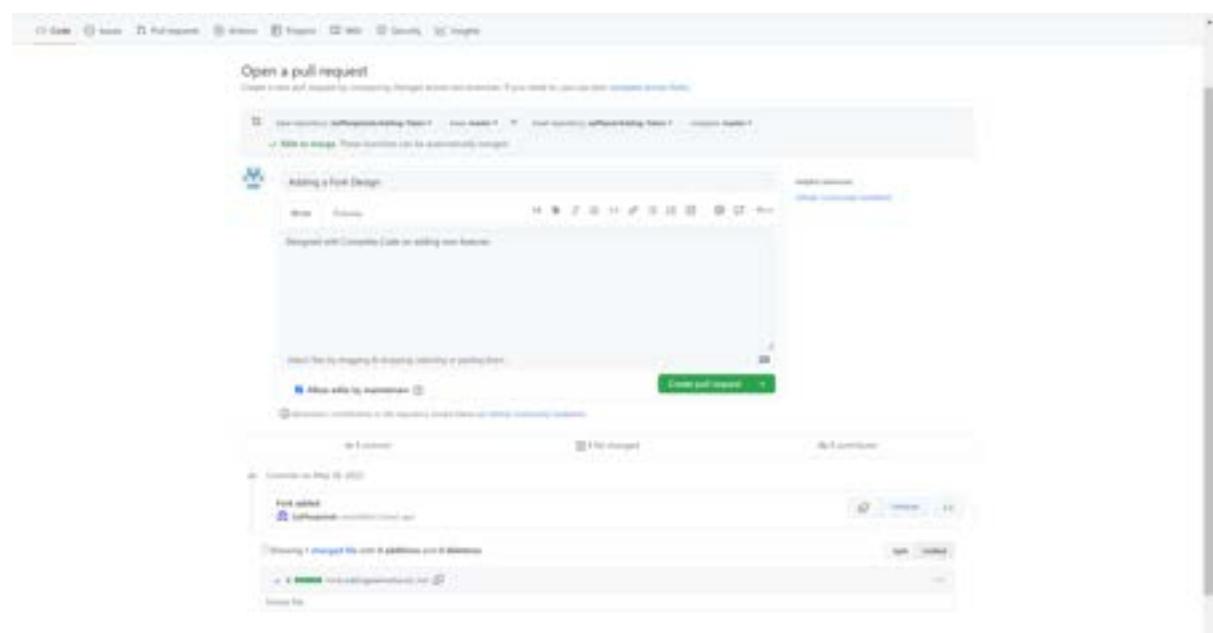


Fig. open pull request

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

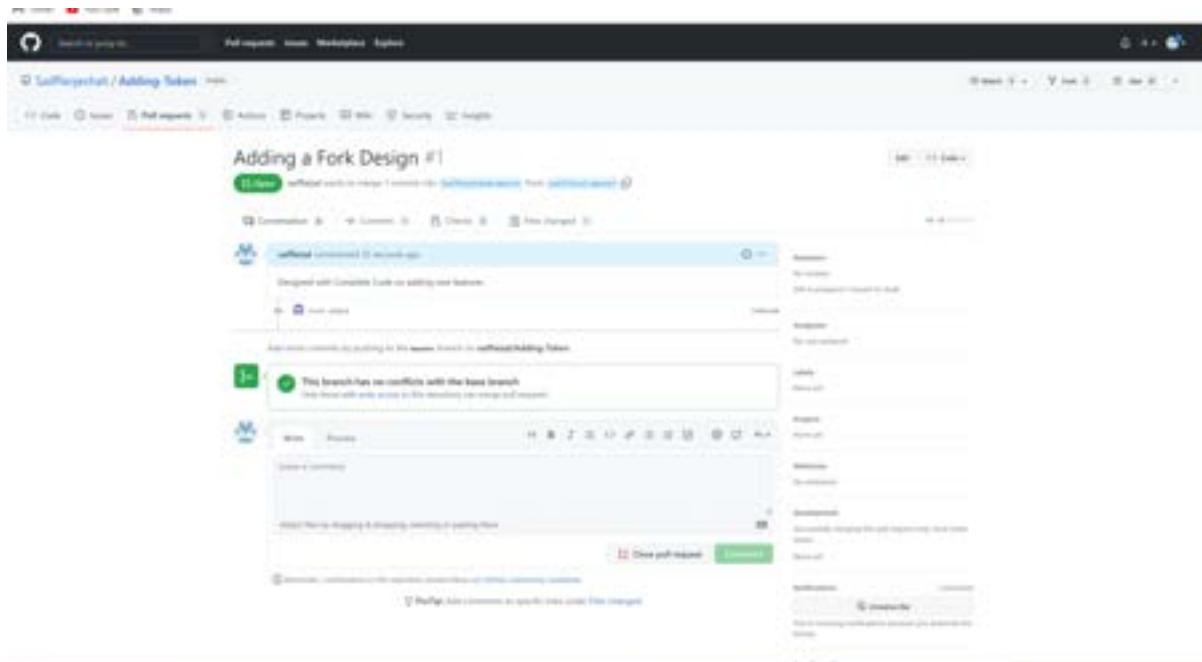


Fig. open pull request successful

Now, to view this pull request open owner organizational account

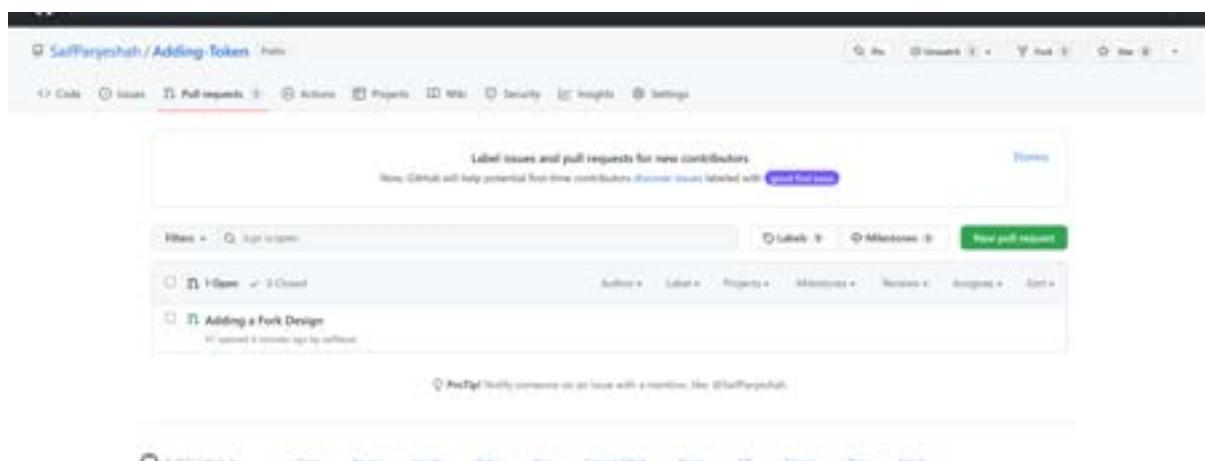


Fig. Owner view the pull request

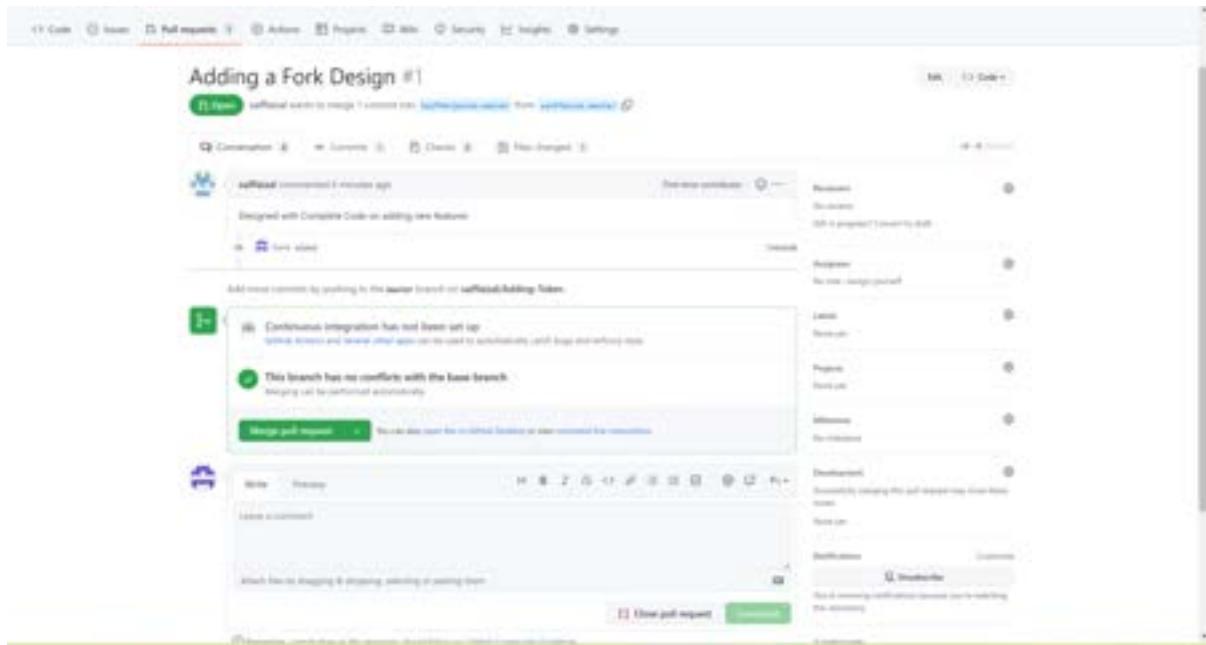


Fig. Merging or closing pull request

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

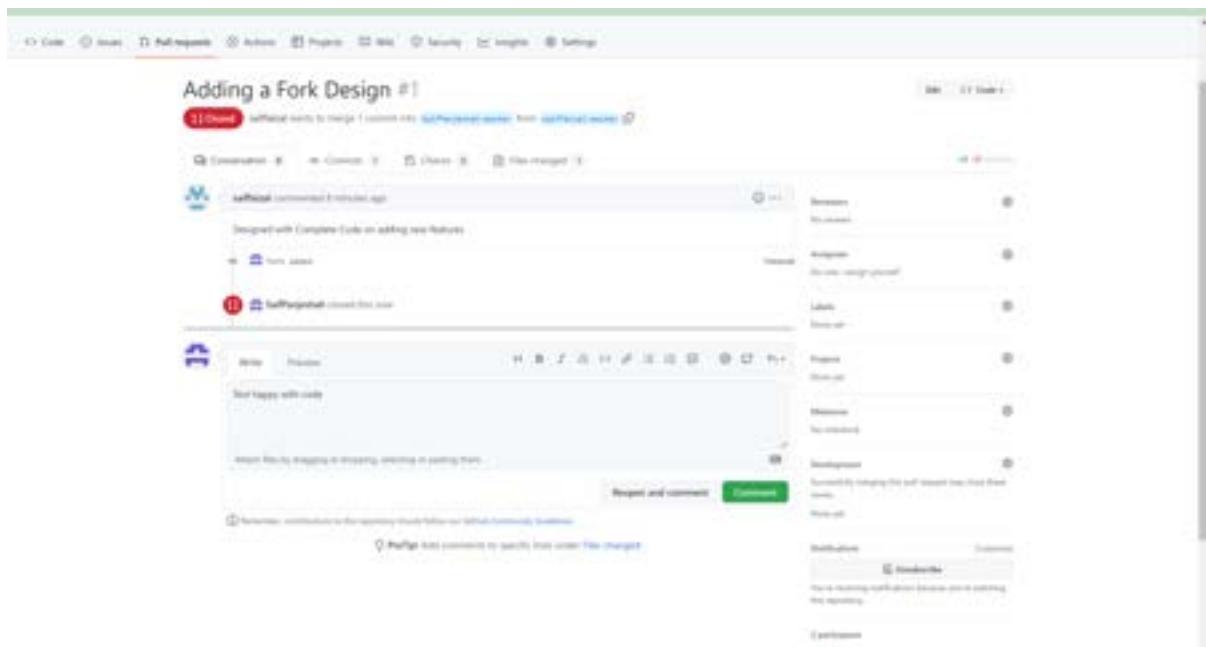


Fig. Owner closing this pull request

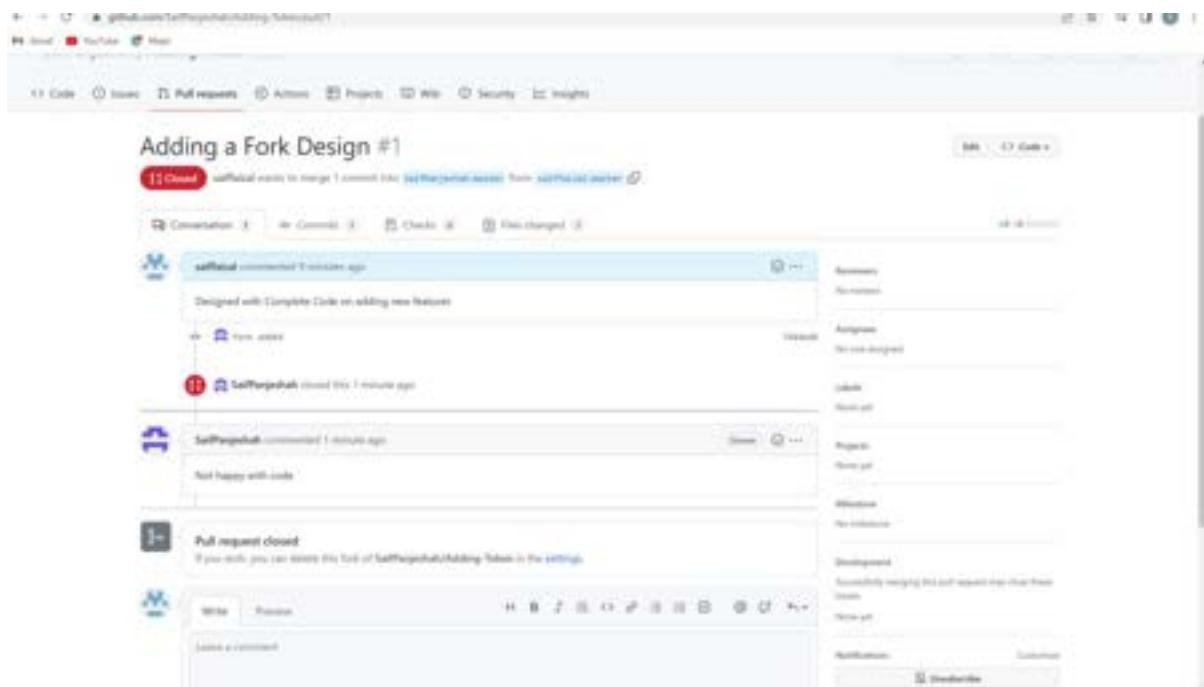


Fig. Closed Successful view from Member account

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

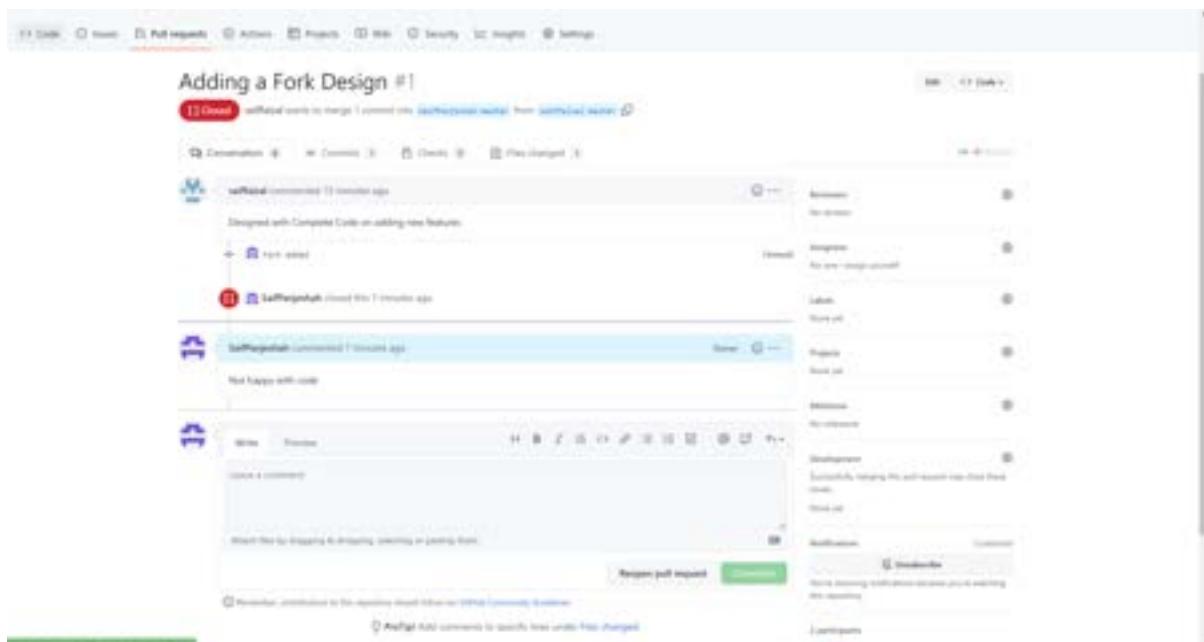


Fig. reopen the pull requests

Merging the pull requests:

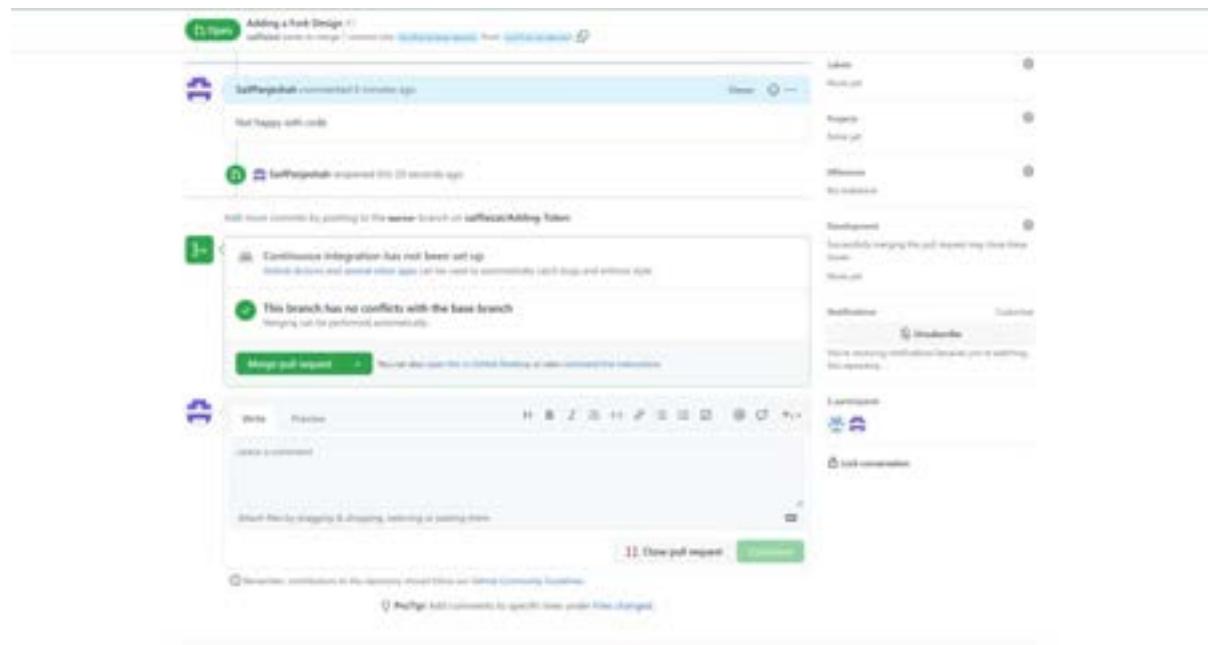


Fig. Merging the pull requests

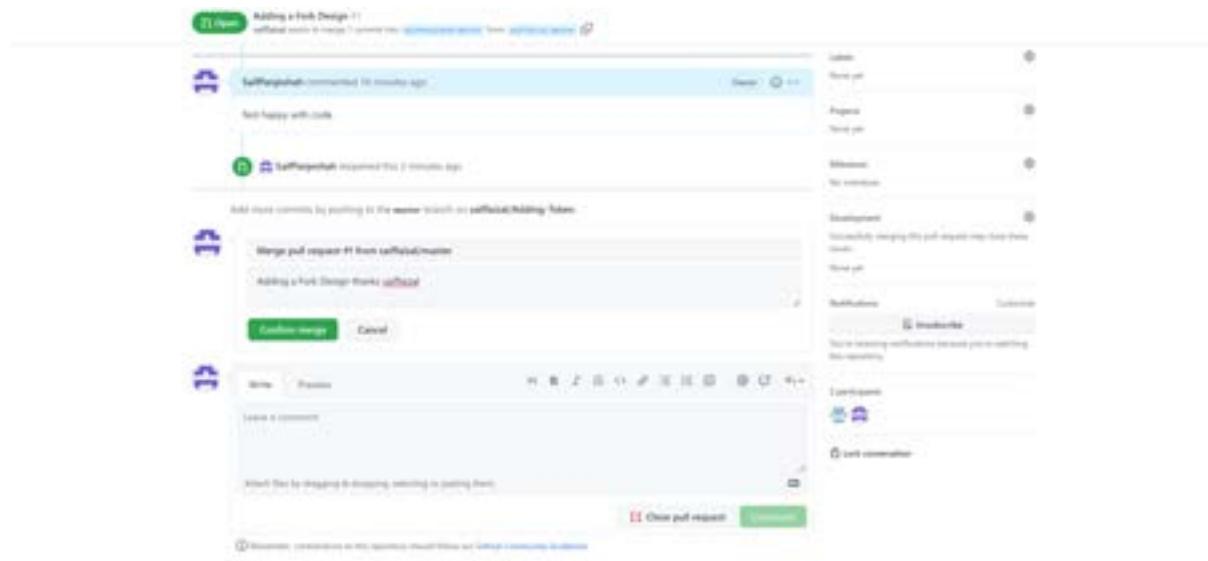


Fig. Confirm merging

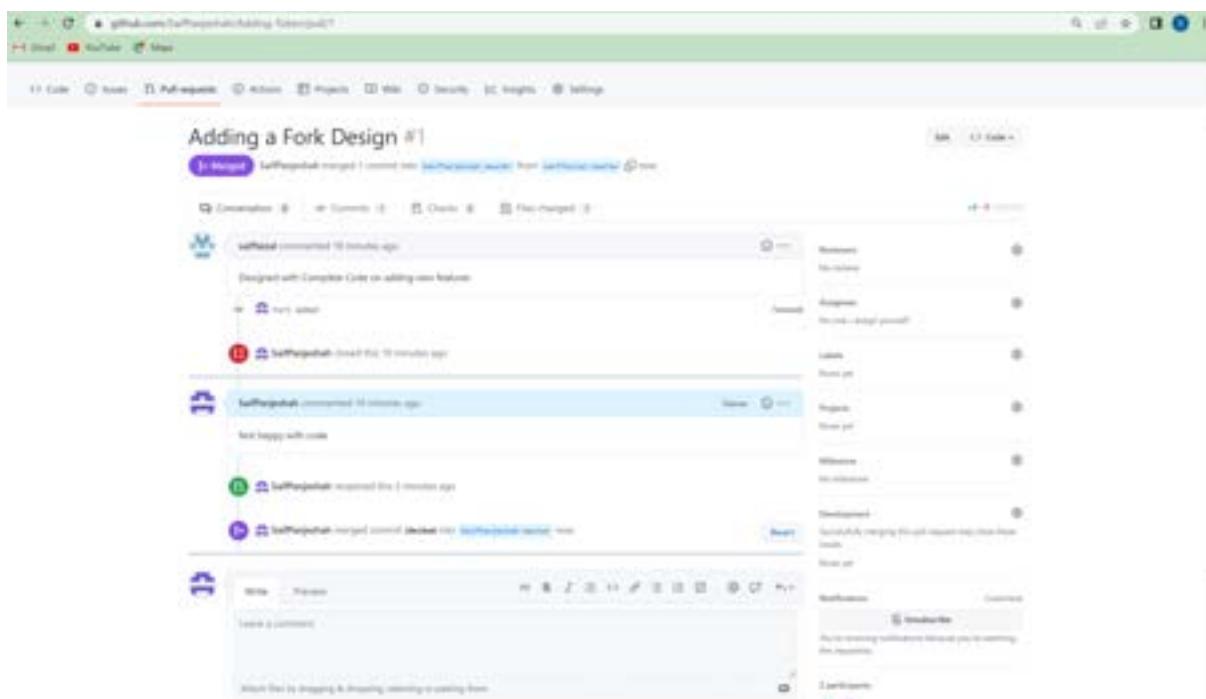


Fig. Merge successful

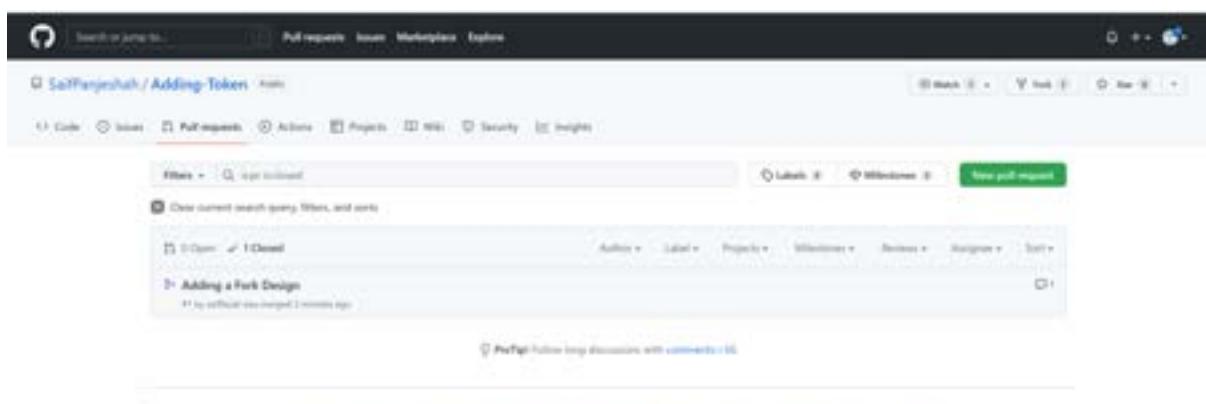


Fig. Close and successful Merge view from member account

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.**

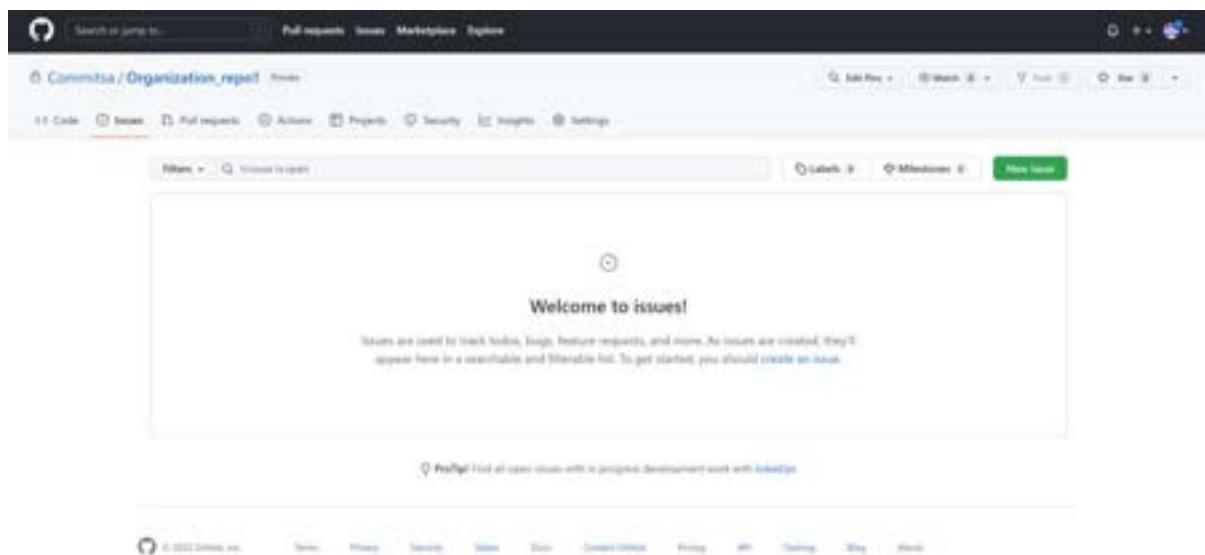


Fig. Creating a new Issue on organizational account repository

- 2. Click New issue:**

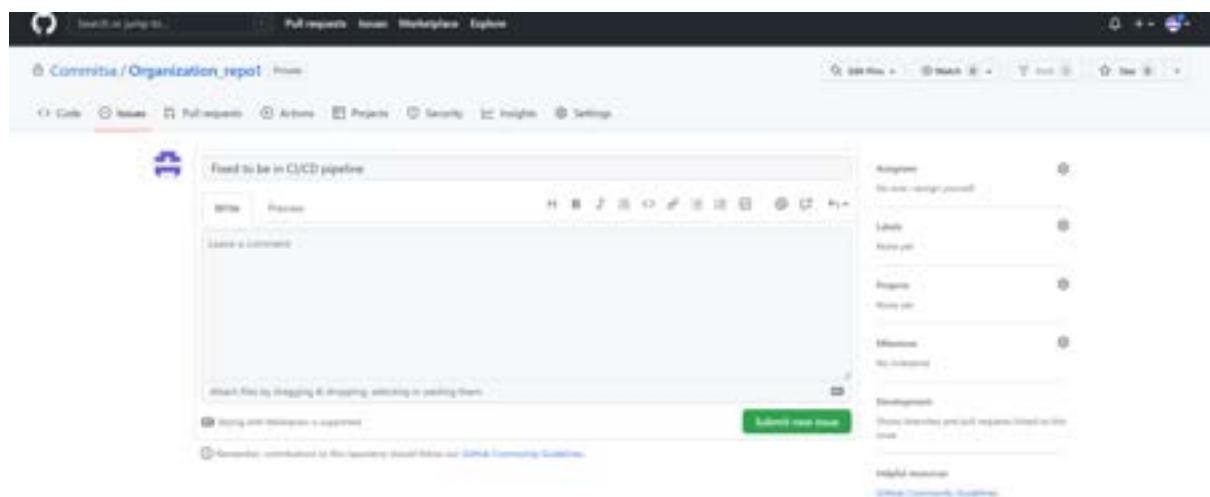


Fig. Creating Issues with CI/CD Pipelines of Devops

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

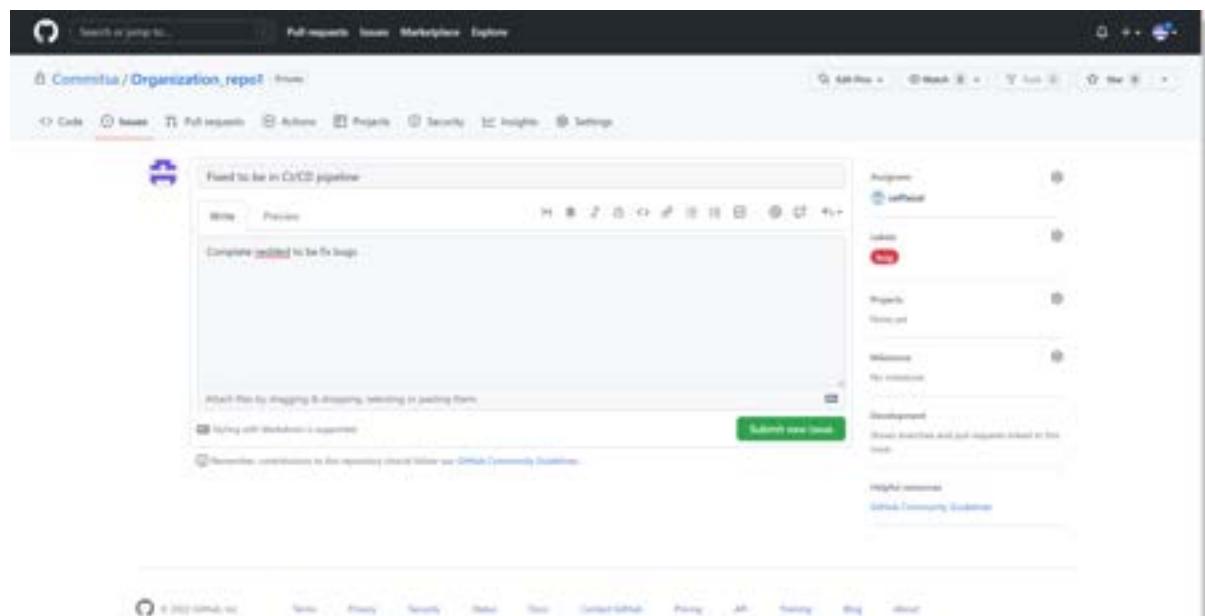


Fig. Fixed needs to be done

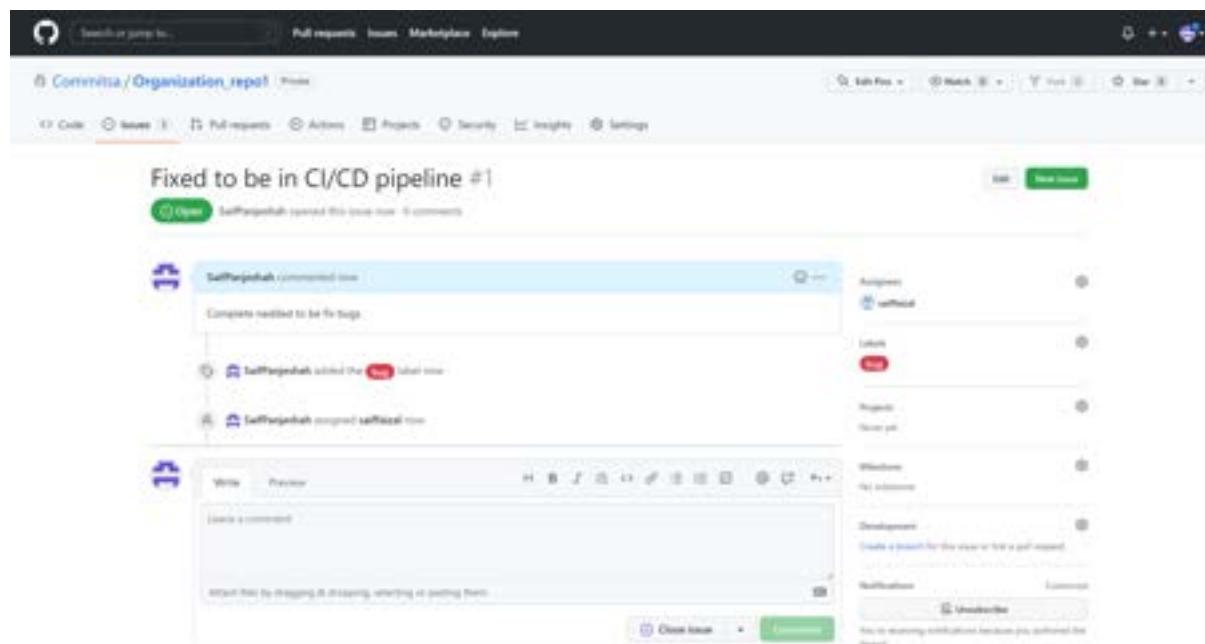


Fig. Issue successful Created

Closing the Issue:

The screenshot shows a GitHub organization repository interface. At the top, there are navigation links for 'Code', 'Issues' (which is selected), 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. Below the header, the repository name 'Committee / Organization_repo1' and the word 'Issues' are displayed. A search bar and filter options ('1 Open', '0 Closed') are present. A specific issue titled 'Fixed to be in CI/CD pipeline' is highlighted with a red label. The issue details show it was opened by 'selfDepend' 3 minutes ago. The issue body contains the text 'Complete modified to be fix (sage)'. On the right side of the issue card, there are buttons for 'Edit', 'New issue', and other repository management options.

Fig. view on member account

This screenshot shows the same GitHub issue page as above, but now the member 'selfDepend' has responded to the issue. The comment reads: 'I fixed this issue in my local branch please check'. The member has attached their local branch commit history, which includes the commit message 'Complete modified to be fix (sage)' and the addition of a 'solved' label. The right sidebar of the issue page shows standard GitHub issue tracking information like labels, milestones, and assignees.

Fig. Member Fixed this issue

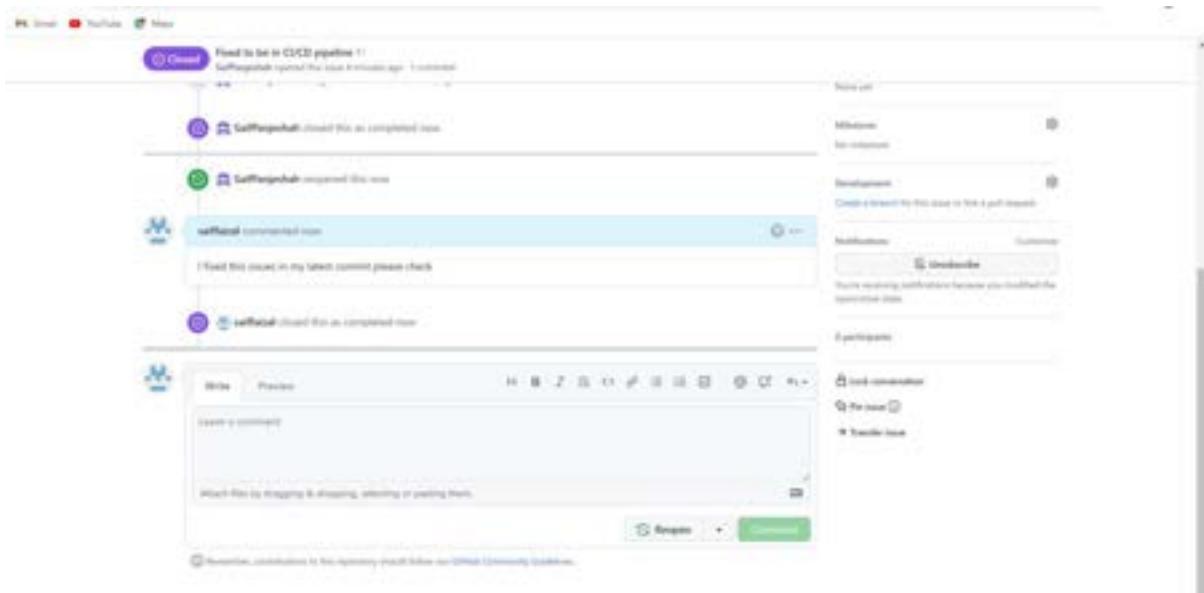


Fig. successful added close and fix commits

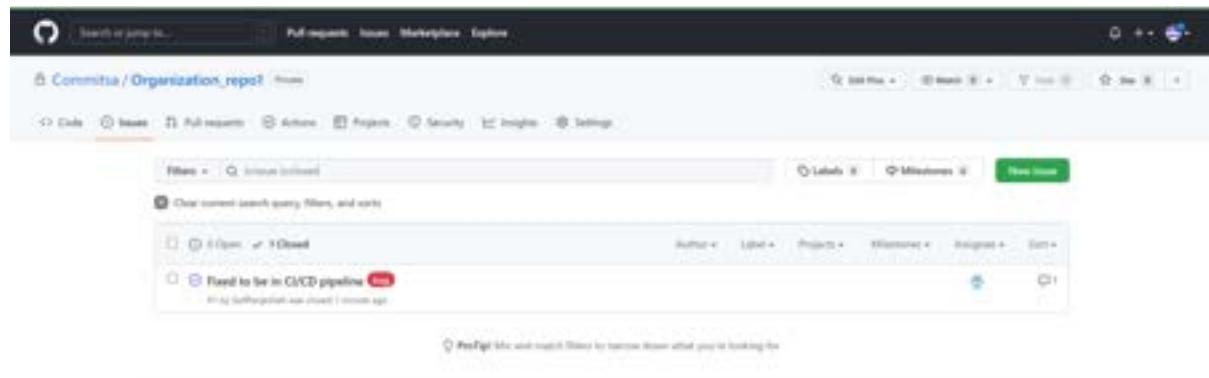


Fig. Successful close this Issue

Working with GitHub Projects:

Let's Explore the Project ribbon in GitHub:

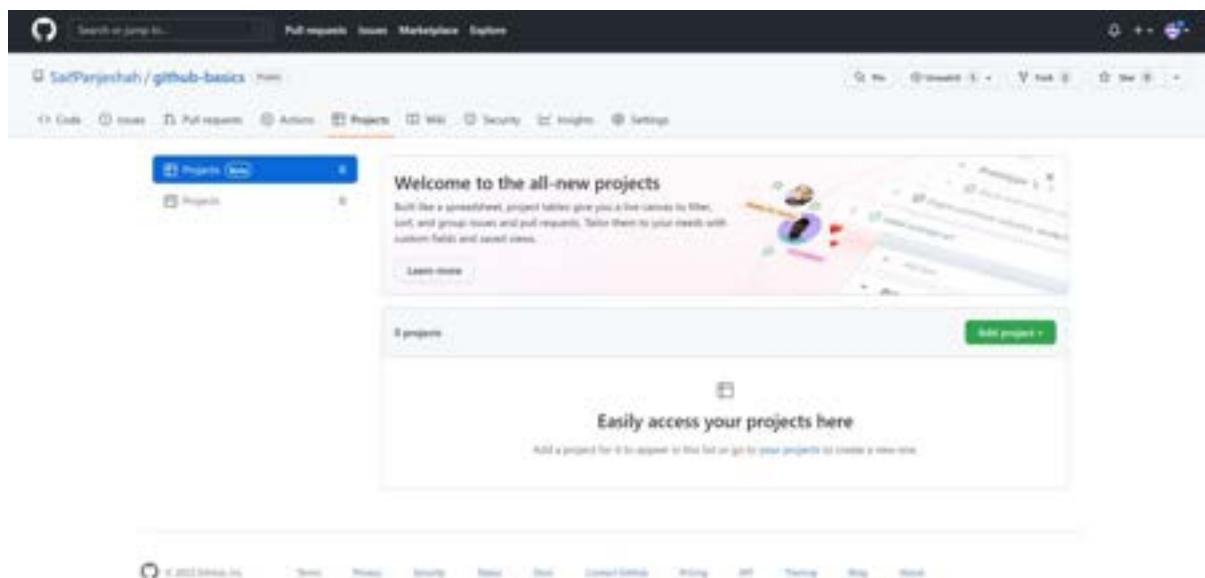


Fig. Project in GitHub

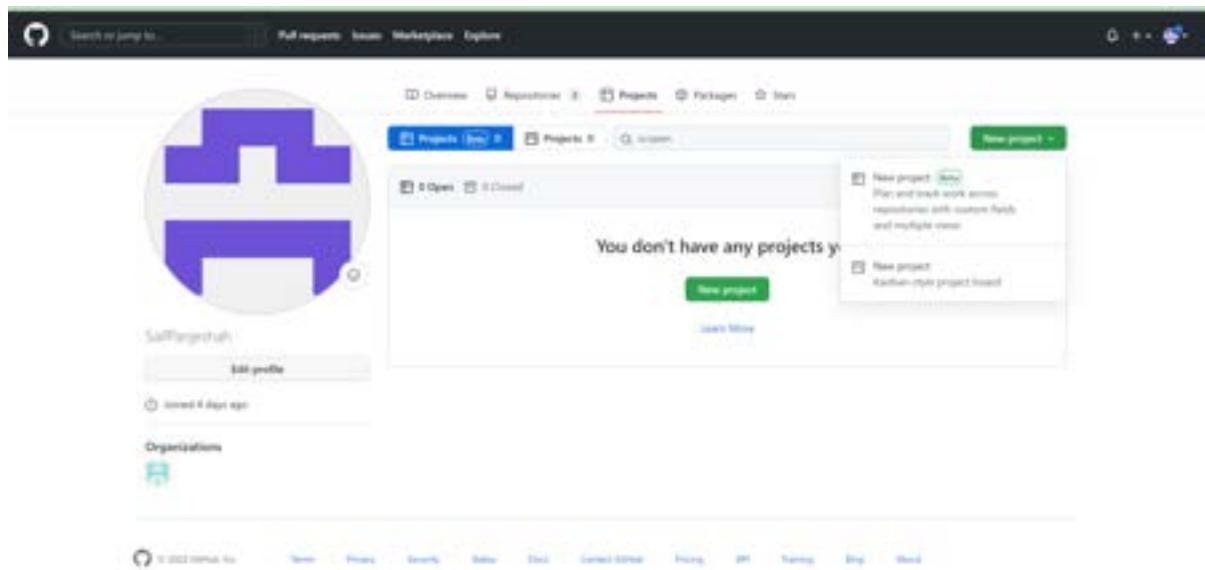


Fig. Creating a Project with Kanban Style

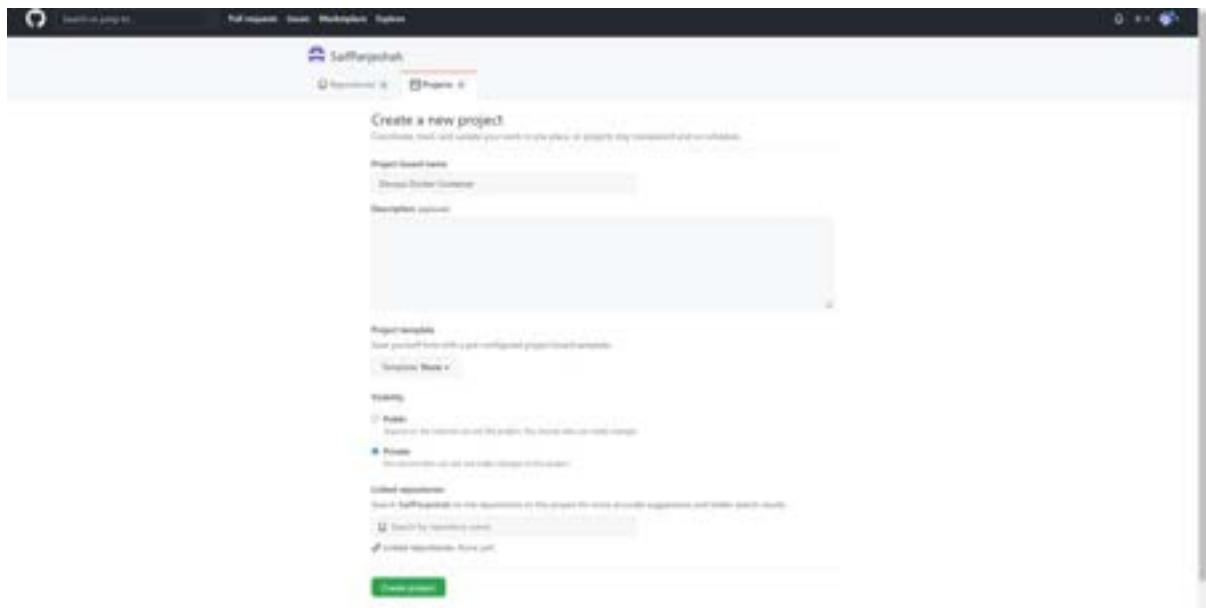


Fig. Devops Docker Container Project

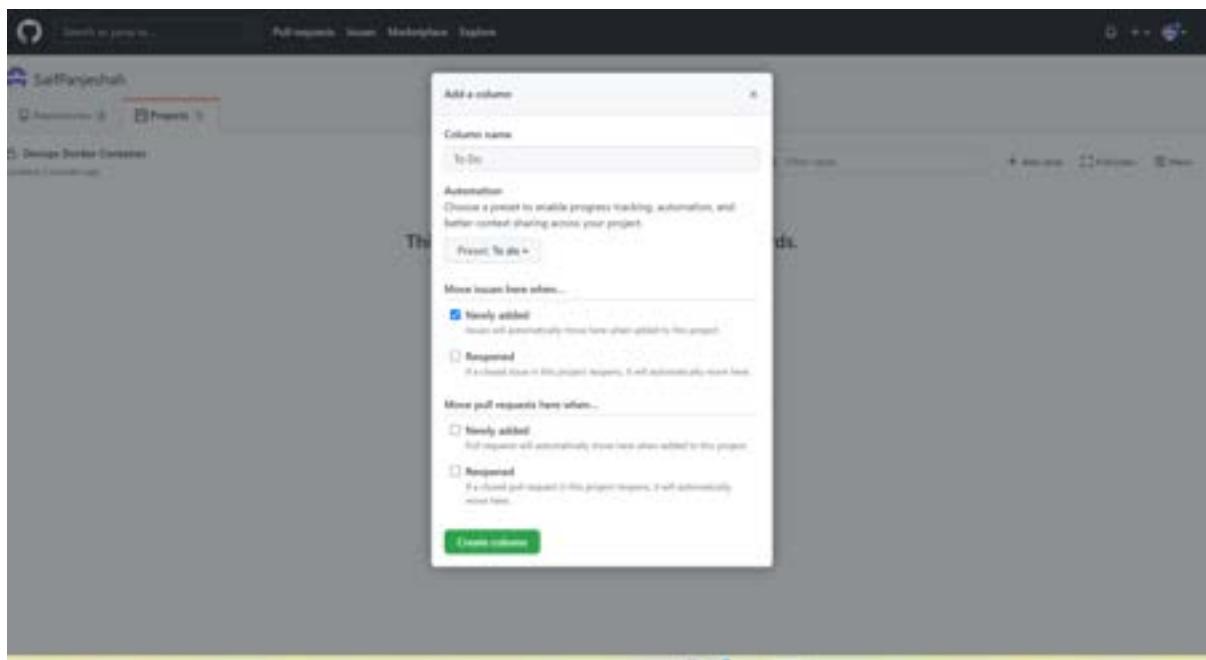


Fig. adding a column what to do

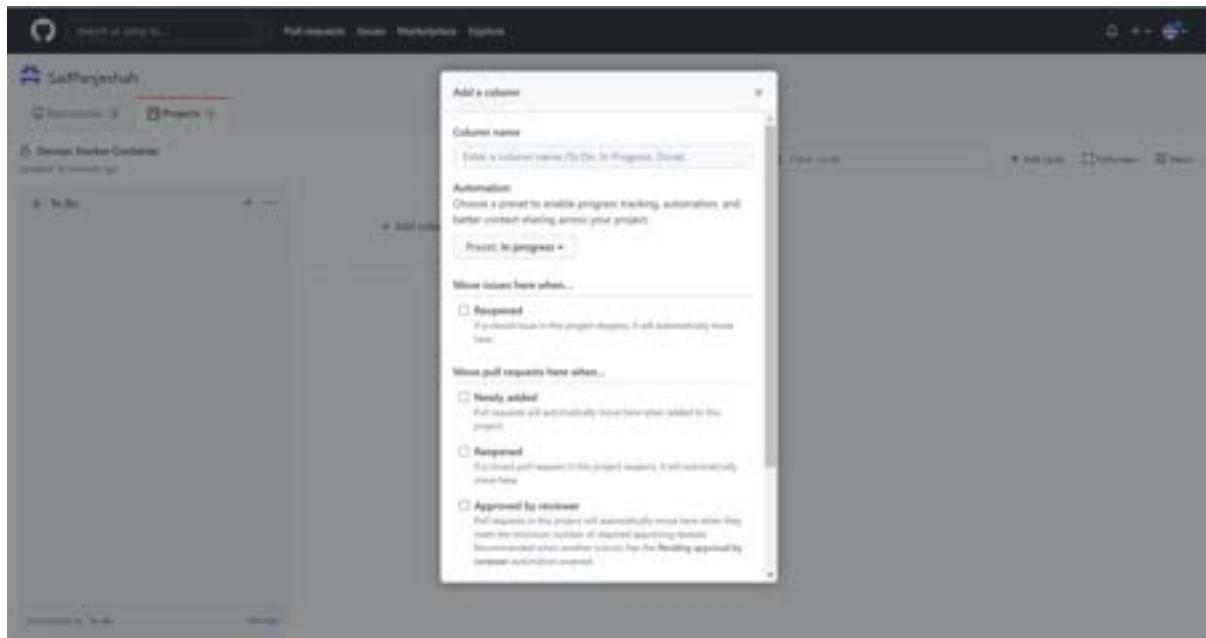


Fig. Creating another column with InProgress

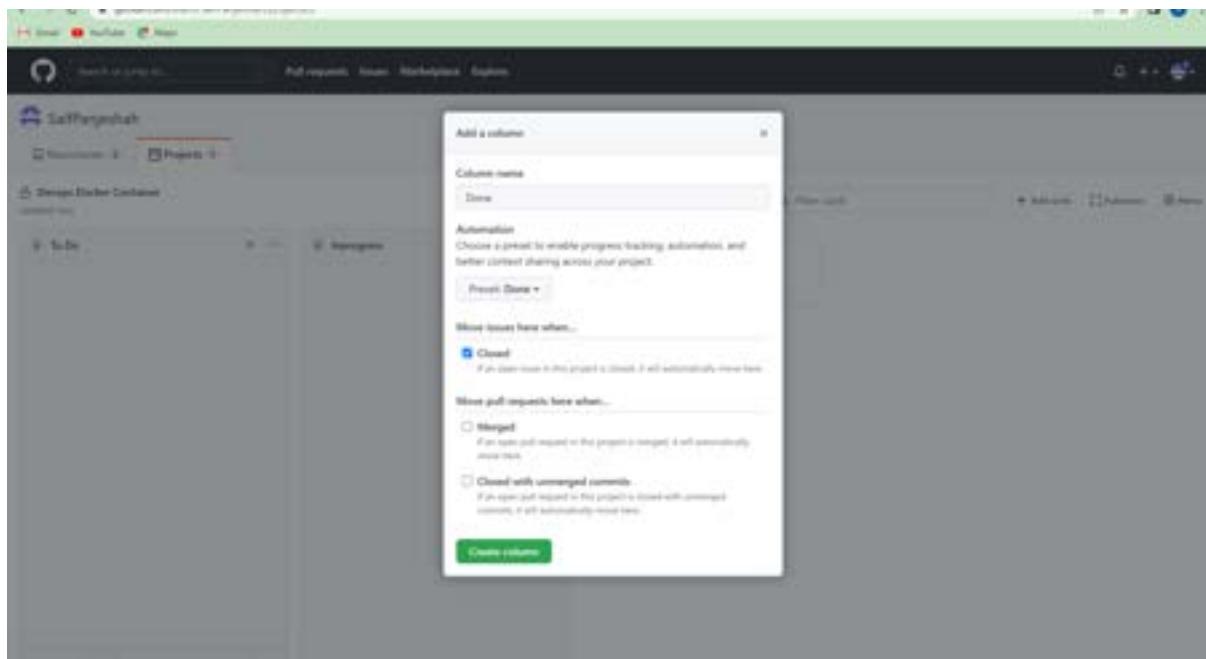


Fig. Final Column with work Done

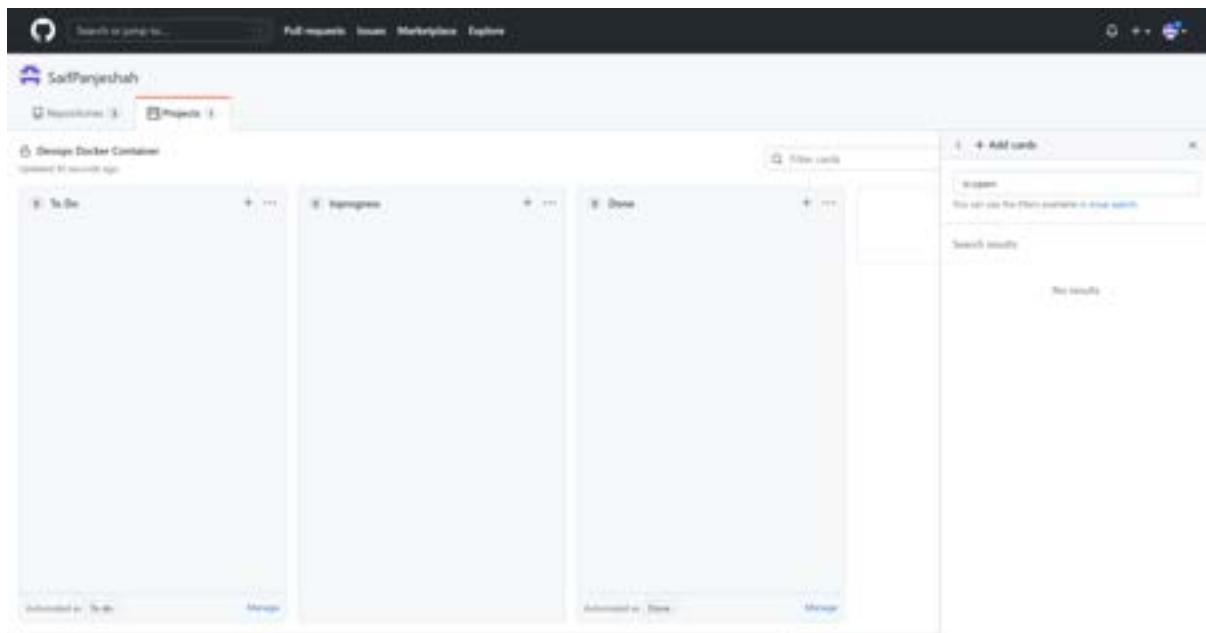


Fig. Project Created successful

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

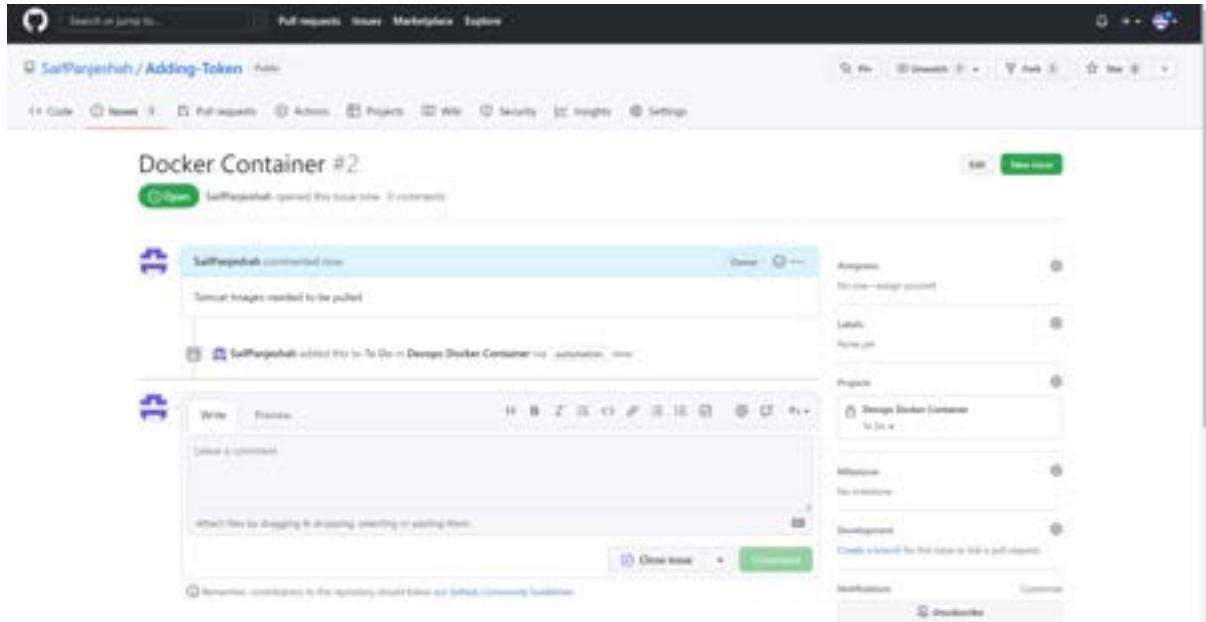


Fig. Creating new Issues

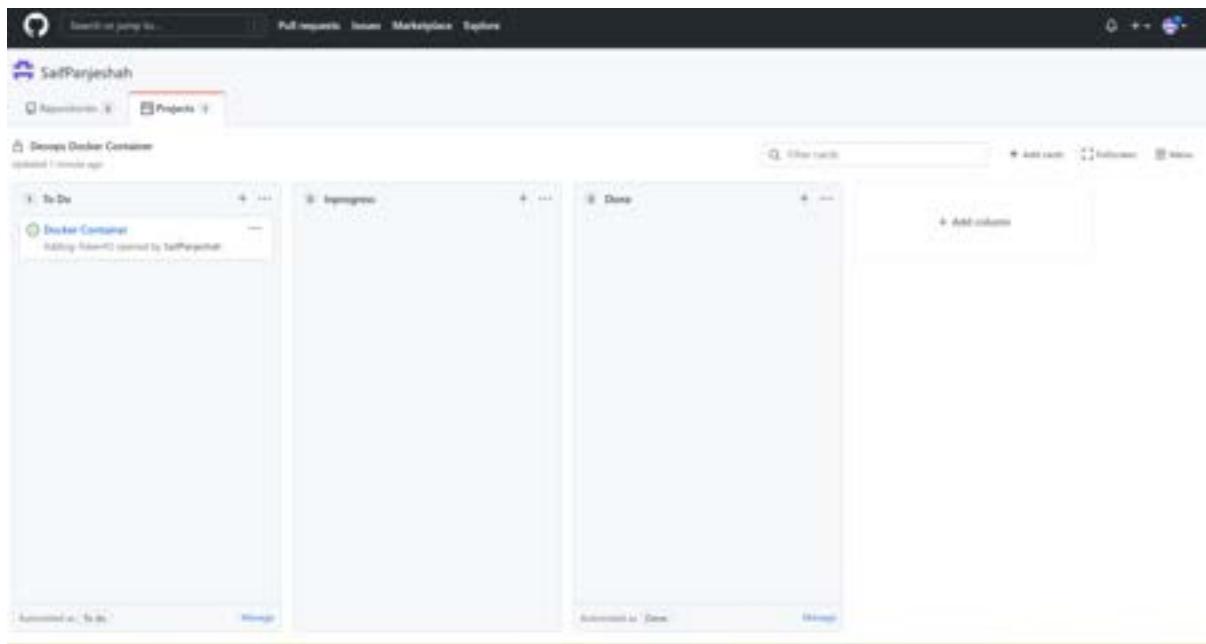


Fig. Card has been successful added to projects

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

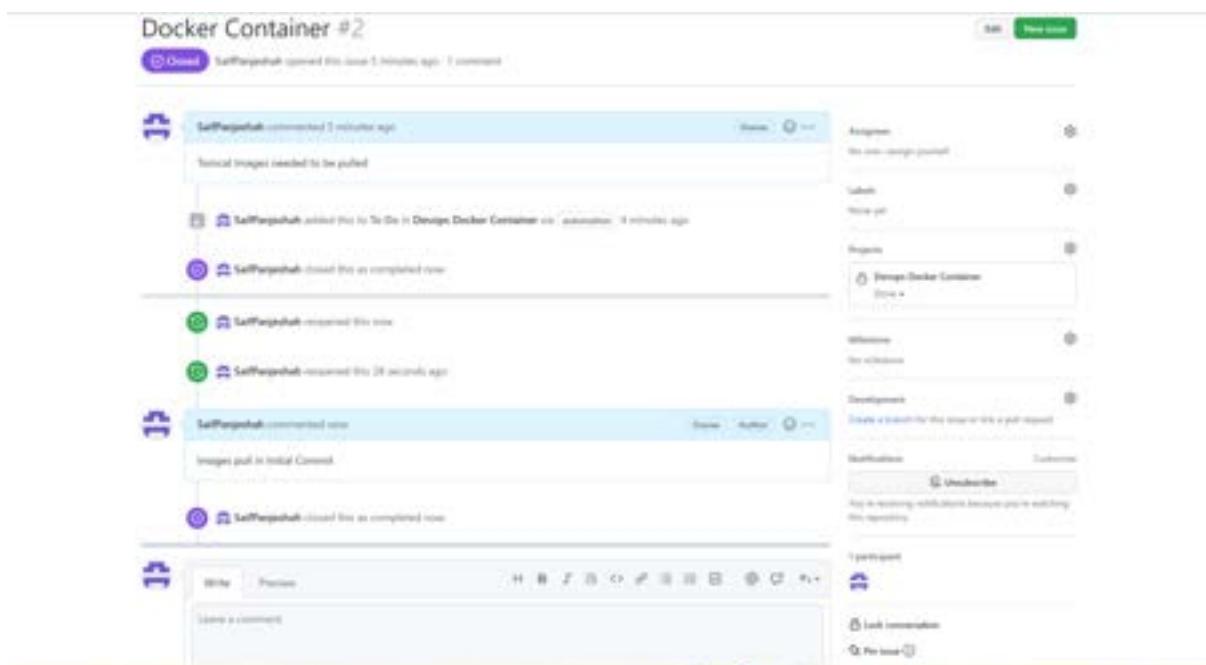


Fig. Closing Issue

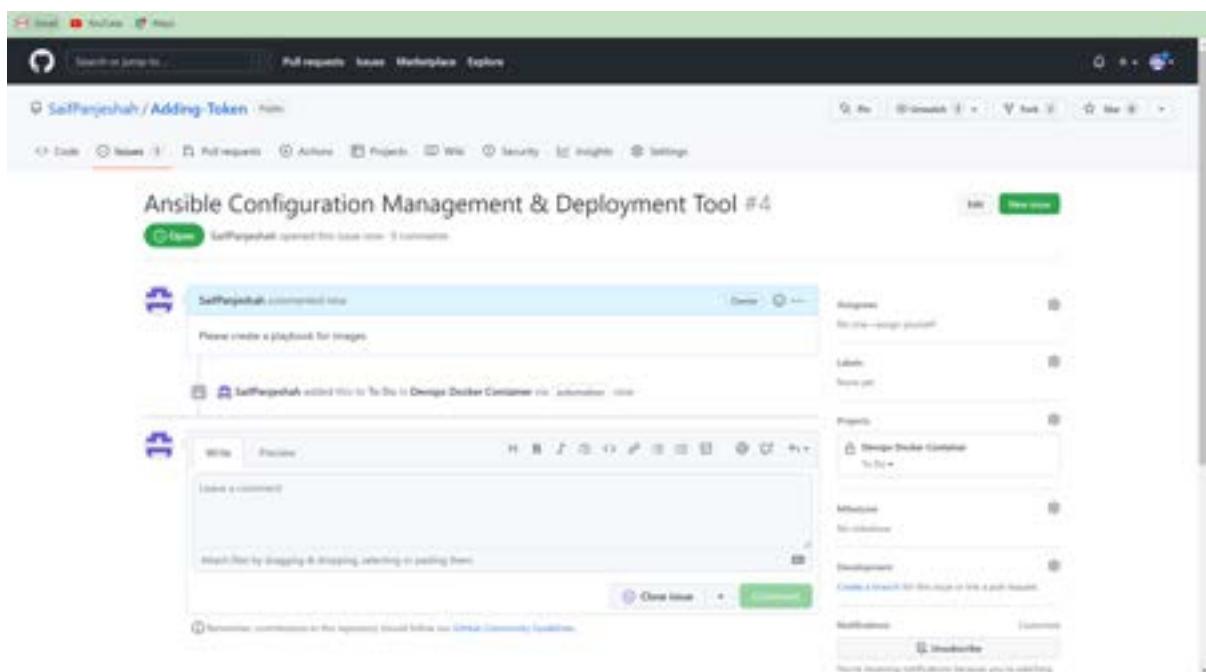


Fig. Creating New Issue

Let's Explore Devops Docker Container Project:

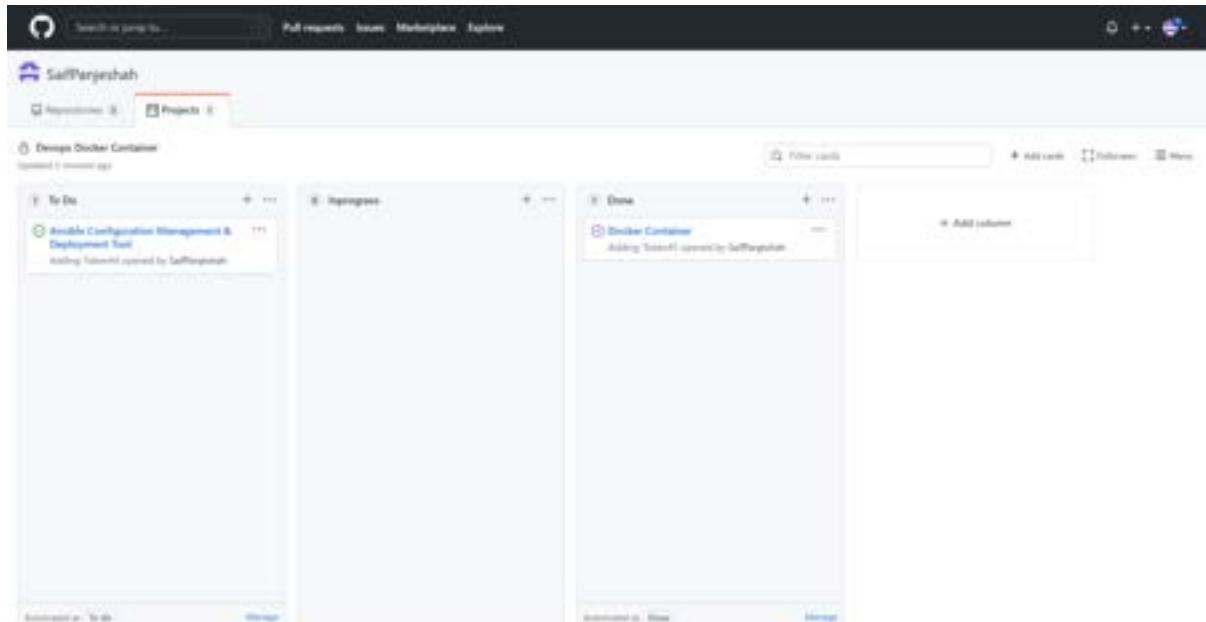


Fig. cards successful created on Devops Docker Container Project

Creating a README File:

README (as the name suggests: "read me") is the first file one should read when starting a new project. It's a set of useful information about a project, and a kind of manual. A README text file appears in many various places and refers not only to programming.



Fig. Creating a README File

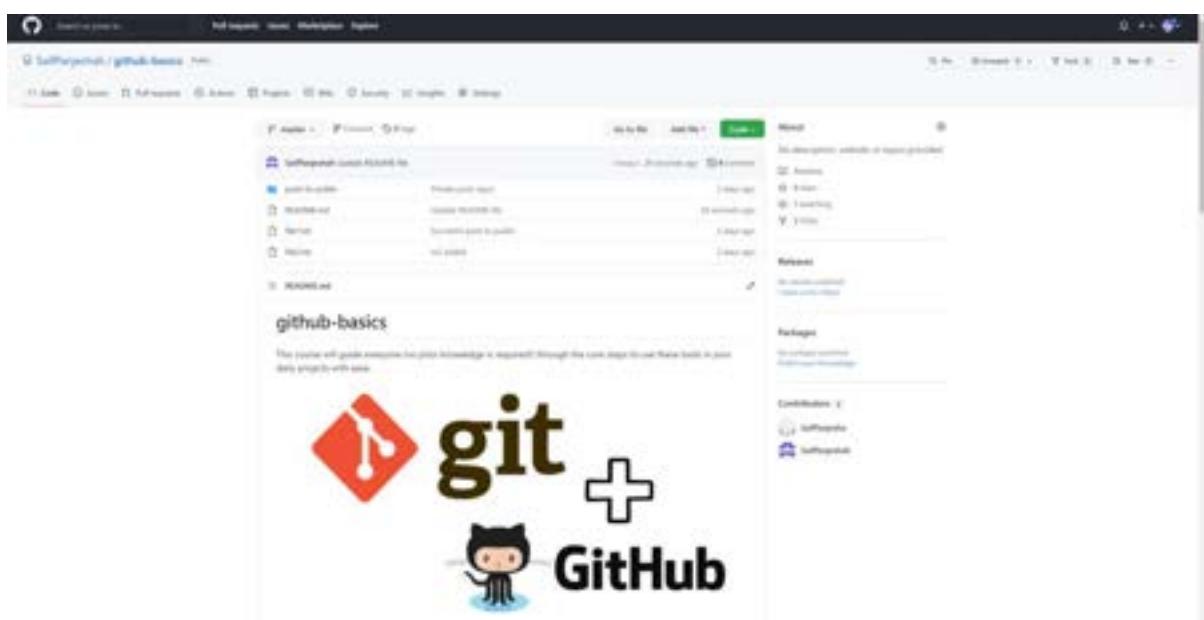


Fig. Complete README File added

Presenting Yourself as a Developer on GitHub:

Making GitHub Page more attractive.

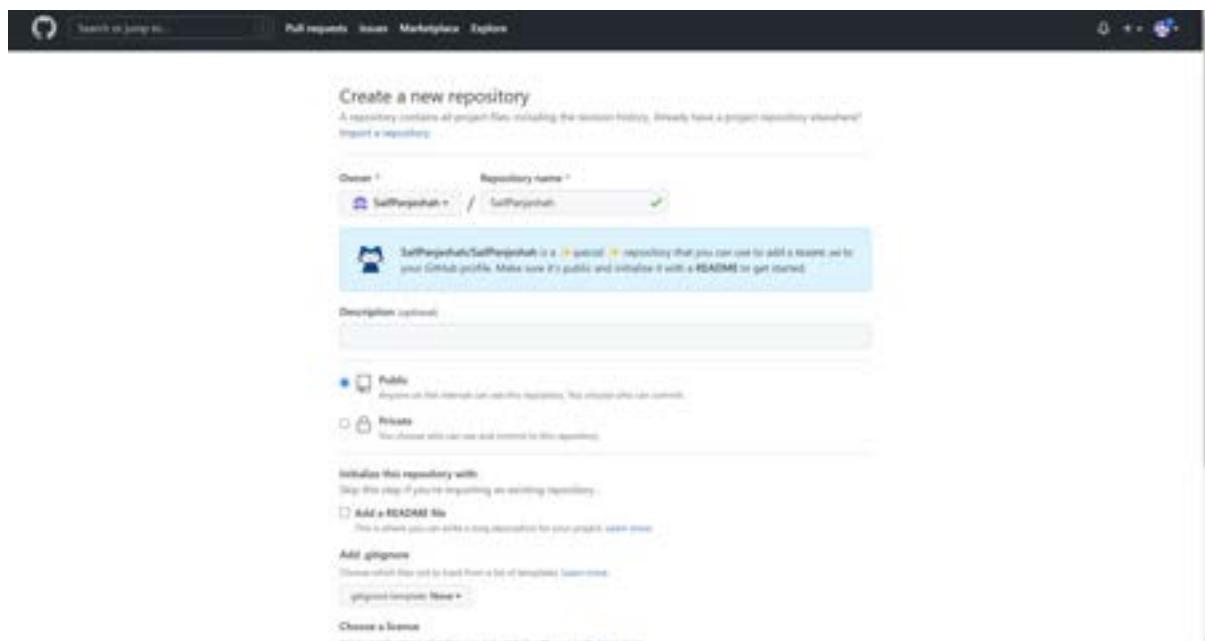


Fig. Creating Special secret repository with owner name

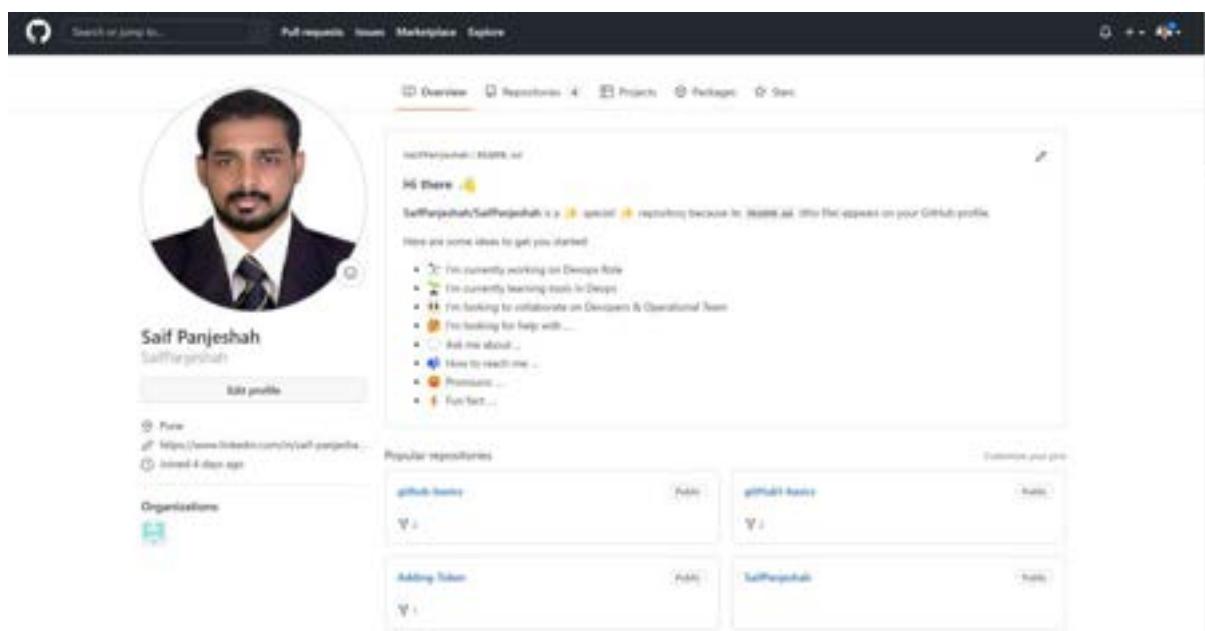


Fig. Presenting Yourself as a Developer on GitHub

About GitHub Stars:

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

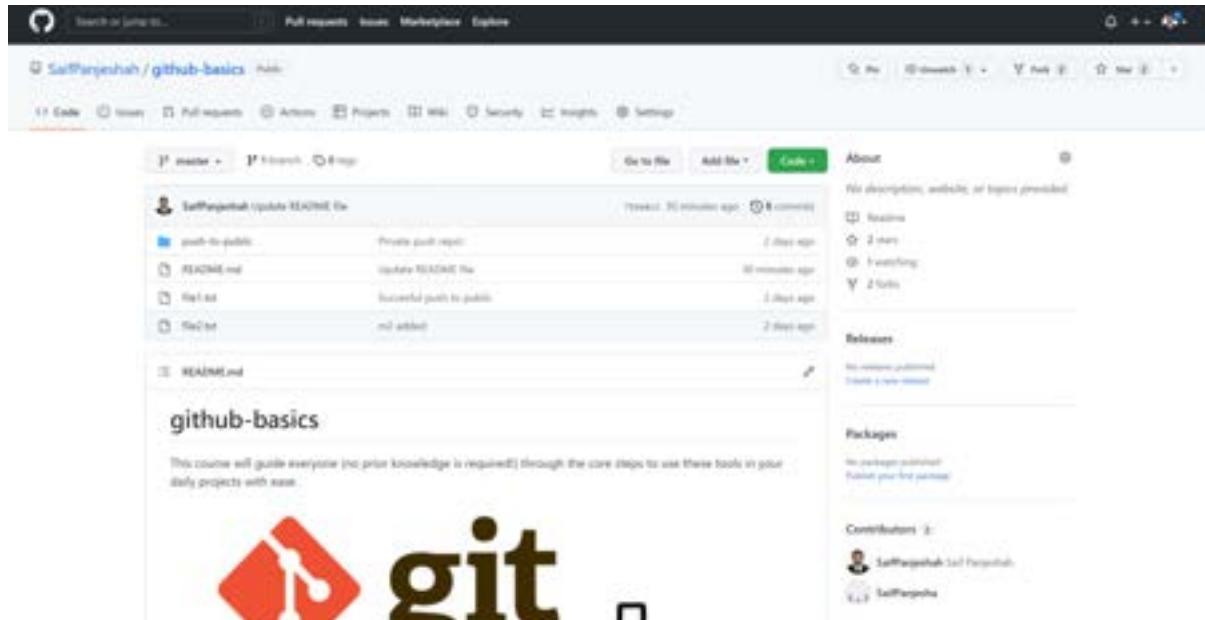


Fig. GitHub Stars

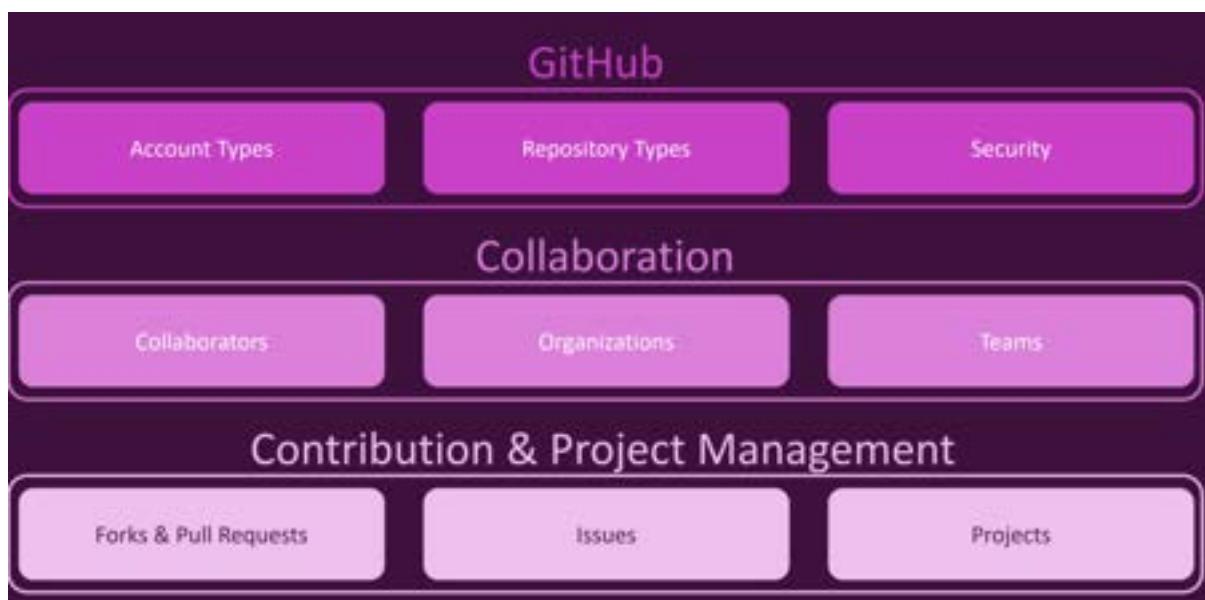


Fig. Wrap up Model Summary

Useful Resources and Link:

More about Permission Levels for User Account Repositories

=> <https://docs.github.com/en/github/setting-up-and-managing-your-github-user-account/managing-user-account-settings/permission-levels-for-a-user-account-repository>

VueJS GitHub Page => <https://github.com/vuejs/vue>