



Git & GitHub

Git is a widely used **distributed version control** and source code management system. It effectively tracks changes to source code, enabling effortless branching, merging, and versioning.

Git

Version Control System is a tools that helps to track changes in code

Git is a Version Control System. It is :

popular

free & Open Source

fast & scalable

It was developed by Linus Torvalds in 2005 for Linux kernel development and it is used for keeping track of code changes and collaborating with others on code.

It uses a decentralized model where each developer has their own copy of the repository and works immediately on the project.

Git manages the projects with repositories and can clone a project to operate locally on it.

With staging and committing it track changes and control.

You can pull the latest code of the project to the local copy, and push local updates to the main projects.

Mainly used for:

- Tracking the history
- Collaborate

Why Use Git?

You need to know that around 70% of developers worldwide use Git for development. Some of the prominent reasons for using Git are:



- Developers can work together from anywhere.
- Developers can see the full history and can compare the previous and new changes of the project.
- Developers can retreat to earlier versions of a project.

Github

Website that allows developers to store and manage their code using Git.

<https://github.com>

We create upload our data in a folder structure on GitHub. These folders are known as Repository (Repo).

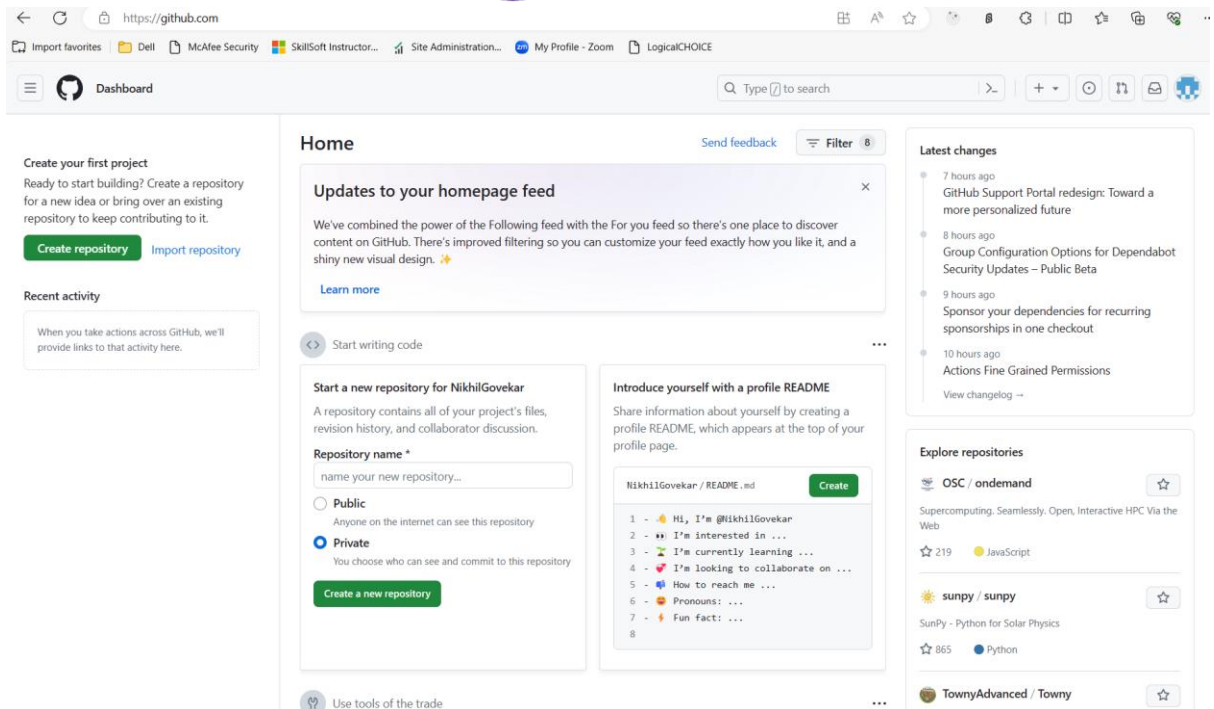
GitHub is a hosting service for Git repositories and if you have a project hosted on GitHub, you can access and download that project with commands on any computer you have access and make your changes and push the latest version back to GitHub.

GitHub allows you to store your repo on their platform.

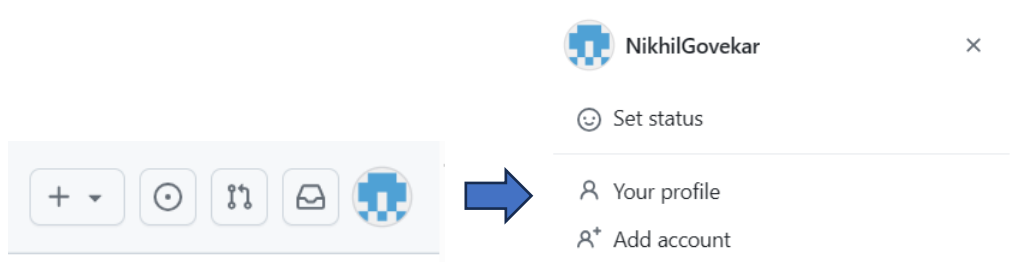
It is also comes with GitHub, ability to collaborate with other developers from any location.

Create GitHub Account:

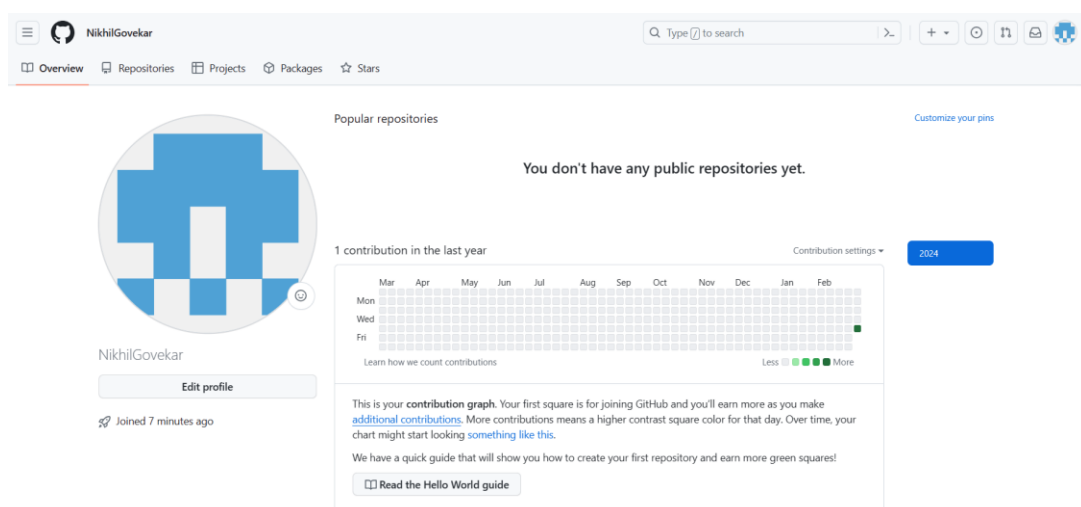
- Go to <https://github.com> and create your account
- Once the account is created you will see the dashboard



- Go to right top corner to select your profile:

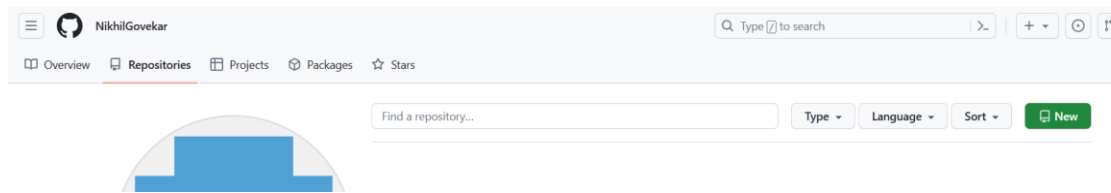


- Click on Your Profile:





- Click on Repositories → New button to create new repo i.e. new project:




- Add below details to create new Repo → Project-Demo

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner * Repository name *



 NikhilGovekar / Project-Demo

✔ Project-Demo is available.

Great repository names are short and memorable. Need inspiration? How about [shiny-fishstick](#) ?

Description (optional)

This is my first project

- ☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.
- ☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

- ☒ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: **None**

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: **None**

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

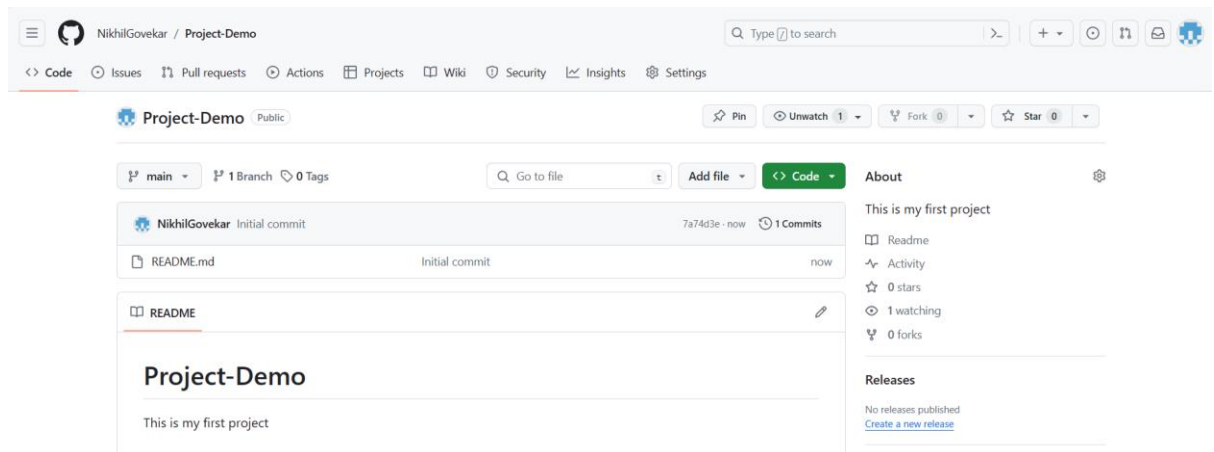
This will set  **main** as the default branch. Change the default name in your [settings](#).

 You are creating a public repository in your personal account.

Create repository

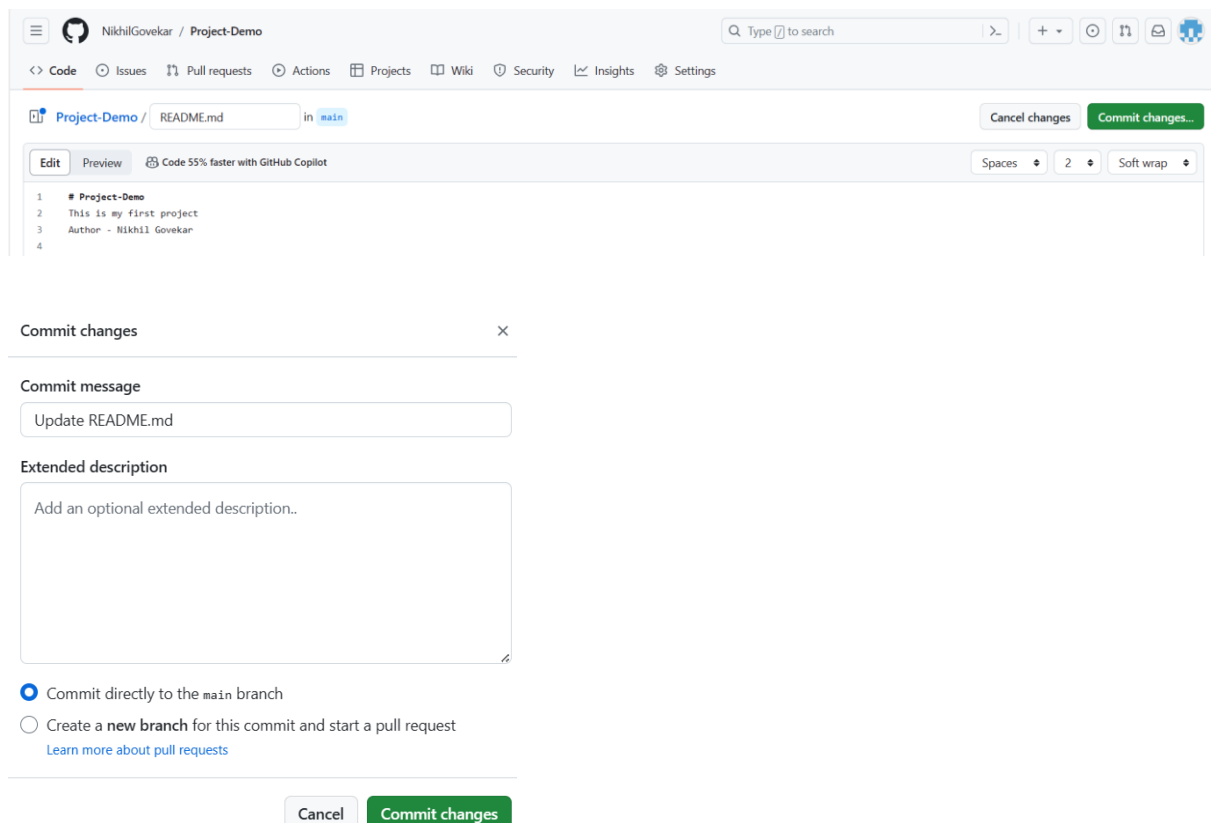


- We can see our Repo → Project-Demo is created.



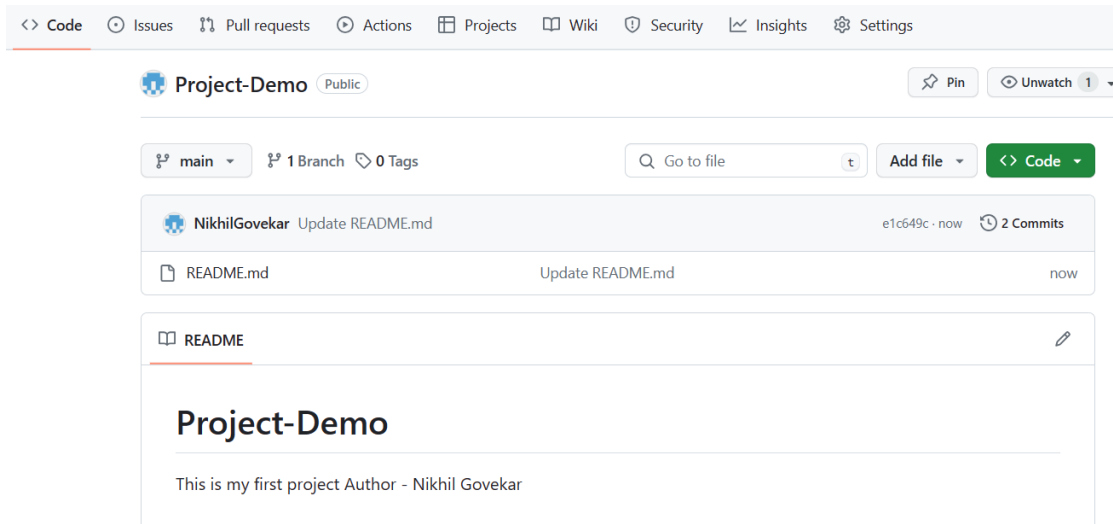
- Changes in the Repo → This is a two-step process called as “add and commit”. In simple words, commit the changes means take a screenshot and store it in memory for track the changes.
- Let's make some changes in Readme file. Edit ReadMe file →

Add → Author – Nikhil Govekar → Click on Commit changes.. button

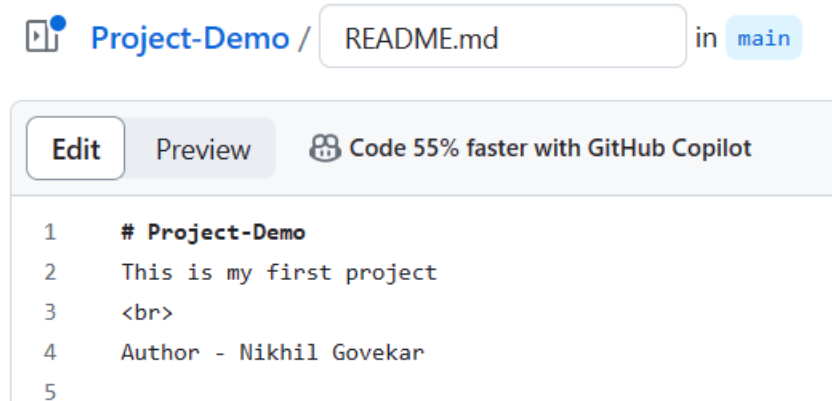




- Click on Commit Changes.
- Click on <> Code option and see the changes:



- You will see the committed changes.
- To make changes in readme file we need to know html.
- Click on Edit option for README file and add
 tag as shown below:



- Commit Changes ...



Commit changes

Commit message

Added Next Line

Extended description

Add an optional extended description..

☒ Commit directly to the main branch

☐ Create a new branch for this commit and start a pull request

[Learn more about pull requests](#)

Cancel Commit changes

- Click on Commit changes button and check the modifications using <> Code option as shown below:

NikhilGovekar / Project-Demo

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Project-Demo Public

main 1 Branch 0 Tags

Go to file t Add file <> Code

NikhilGovekar Added Next Line 13b1028 · now 3 Commits

README.md Added Next Line now

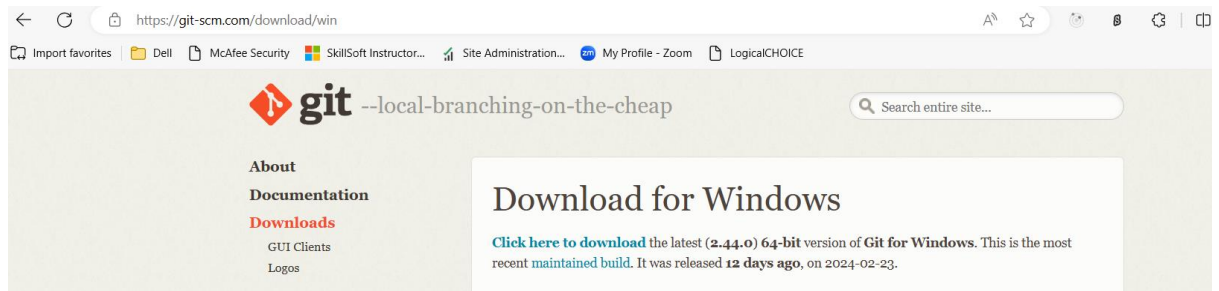
README

Project-Demo

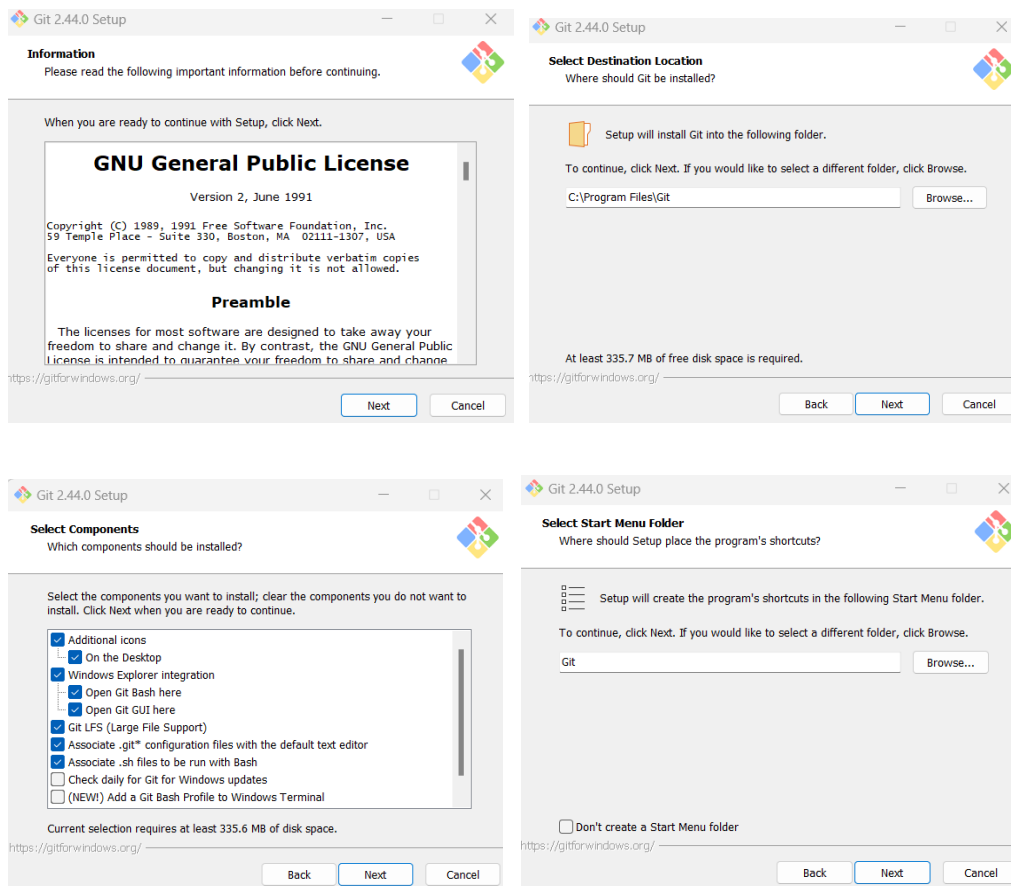
This is my first project
Author - Nikhil Govekar

Setting up Git in the system:

- Download and install
 - Visual Studio Code Editor (VSCode)
 - Windows (Git Bash)
- Command to check version: **git --version**
- Download git – <https://git-scm.com>



- Click on → Click here to download → Once download is completed, run the exe.





The four screenshots show the following steps in the Git 2.44.0 Setup wizard:

- Choosing the default editor used by Git:** The 'Use Vim' option is selected. A note mentions that Vim is the default for Windows only and is recommended to switch to a modern GUI editor.
- Adjusting the name of the initial branch in new repositories:** The 'Override the default branch name for new repositories' option is selected, with 'main' entered in the text field.
- Adjusting your PATH environment:** The 'Use Git and optional Unix tools from the Command Prompt' option is selected.
- Configuring experimental options:** All experimental options are unchecked.

Keep all other options as is... click on Install button.

The 'Completing the Git Setup Wizard' screen displays the Git logo and the following text:

Setup has finished installing Git on your computer. The application may be launched by selecting the installed shortcuts.

Click Finish to exit Setup.

☐ Launch Git Bash

☒ View Release Notes

At the bottom, there is a large 'Finish' button.

- You will see Git Bash Icon on Desktop. Open and check the version:



```
MINGW64:/c/Users/Nikhil

Nikhil@DESKTOP-T7NUQOT MINGW64 ~
$ git --version
git version 2.44.0.windows.1

Nikhil@DESKTOP-T7NUQOT MINGW64 ~
$
```

- Use can use below command on the terminal:
 - ls -
 - clear – clear screen
 - pwd - gives current working directory

```
MINGW64:/c/Users/Nikhil

Nikhil@DESKTOP-T7NUQOT MINGW64 ~
$ pwd
/c/Users/Nikhil

Nikhil@DESKTOP-T7NUQOT MINGW64 ~
$
```

Configuring Git:

Configuring Git

```
git config --global user.name "My Name"
```

```
git config --global user.email "someone@email.com"
```

```
git config --list
```

- Open Git Bash and configure below mentioned changes:

```
Nikhil@DESKTOP-T7NUQOT MINGW64 ~
$ git config --global user.name "NikhilGovekar"

Nikhil@DESKTOP-T7NUQOT MINGW64 ~
$ git config --global user.email "nikhil.govekar@gmail.com"
```

- Once done. View the changes done using below command:

```
Nikhil@DESKTOP-T7NUQOT MINGW64 ~
```



```
$ git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/etc/ssl/certs/ca-
bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=main
user.name=NikhilGovekar
user.email=nikhil.govekar@gmail.com
```

- We can write all the Git commands in the Git bash prompt. However, better way of working is using VSCode to code as well as giving Git commands.
- Create a folder Git_Demo and open the folder in VSCode.
- Open terminal and check the version:

```
PS C:\Git and GitHub\Git_Demo> git --version
git version 2.44.0.windows.1
```

- In case if its giving error → set the Path environment variable:
Go to *Environment Variable*. Click on to the *Path* variable, *Edit* and add it to the path given below:
 - C:\Program Files\Git\bin
 - C:\Program Files\Git\cmd
- Let's explore few imp commands:



Clone & Status

Clone - Cloning a repository on our local machine

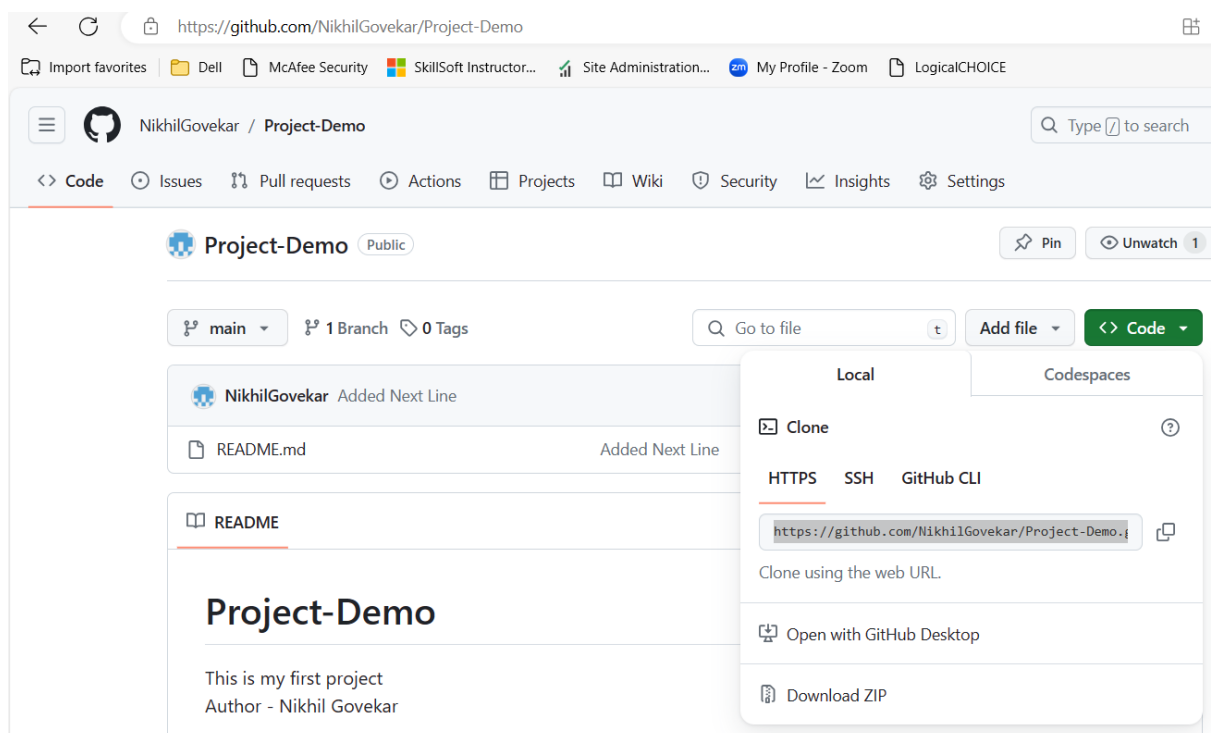
```
git clone <- some link ->
```

status - displays the state of the code

```
git status
```

Note: When we mention remote in command this indicates on GitHub and when we mention local this indicates our machine.

- Let's clone our Repo → Project-Demo from GitHub on our local Machine under GIT_DEMO Folder.

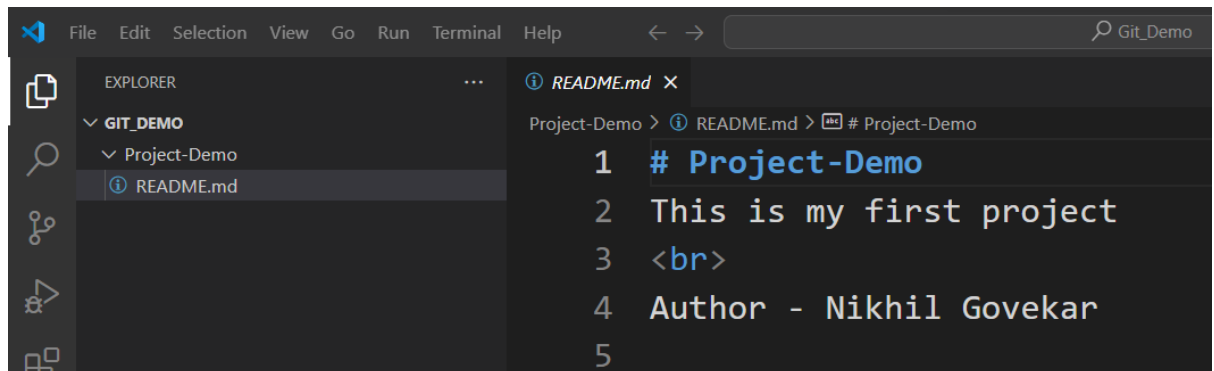


Copy HTTPS url : <https://github.com/NikhilGovekar/Project-Demo.git>

- Give the below command in the terminal to clone the Repo:



```
PS C:\Git and GitHub\Git_Demo> git clone https://github.com/NikhilGovekar/Project-Demo.git
Cloning into 'Project-Demo'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 9 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (9/9), done.
```



- We can see in the Explorer, Project-Demo folder is created.

- Traverse to the respective Project-Demo Folder:

```
PS C:\Git and GitHub\Git_Demo> cd .\Project-Demo\
PS C:\Git and GitHub\Git_Demo\Project-Demo>
```

- To display all files & folder in the Project-Demo folder:

```
PS C:\Git and GitHub\Git_Demo\Project-Demo> ls

Directory: C:\Git and GitHub\Git_Demo\Project-Demo

Mode                LastWriteTime         Length Name
----                -
-a----           07-03-2024   14:14             73 README.md

PS C:\Git and GitHub\Git_Demo\Project-Demo> Get-ChildItem

Directory: C:\Git and GitHub\Git_Demo\Project-Demo

Mode                LastWriteTime         Length Name
----                -
-a----           07-03-2024   14:14             73 README.md
```



- To display all files & folder along with hidden files in the Project-Demo folder:

```
PS C:\Git and GitHub\Git_Demo\Project-Demo> Get-ChildItem -Force

Directory: C:\Git and GitHub\Git_Demo\Project-Demo

Mode                LastWriteTime         Length Name
----                -
d--h--            07-03-2024     14:14         .git
-a----            07-03-2024     14:14        73 README.md
```

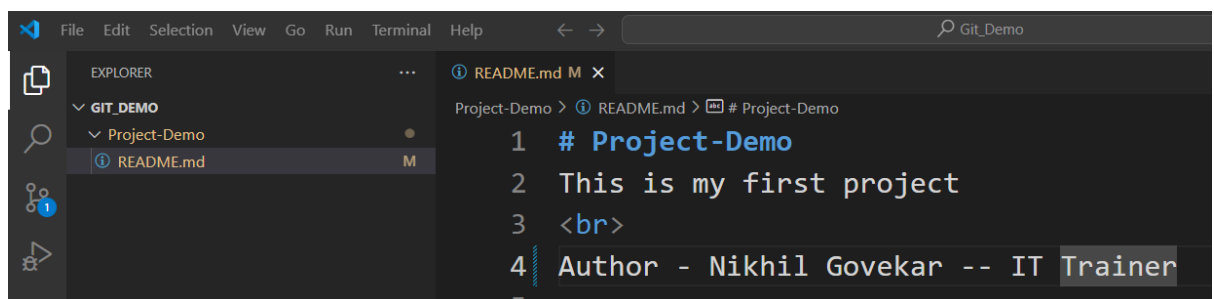
Note: .git folder all file history is tracked and maintained in this folder.

- We use status command to display the status of the code:

```
PS C:\Git and GitHub\Git_Demo\Project-Demo> git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

- Let's modify the readme file in the Project-Demo folder:



- As soon as we modify the file, we can see in Explorer folder and file is highlighted in Yellow and M symbol is shown besides readme file.
- Execute git status command and see the output:



```
PS C:\Git and GitHub\Git_Demo\Project-Demo> git status
On branch main
Your branch is up to date with 'origin/main'.

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:   README.md

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

- Add new index.html file in the Project-Demo folder. Execute git status command and see the output:

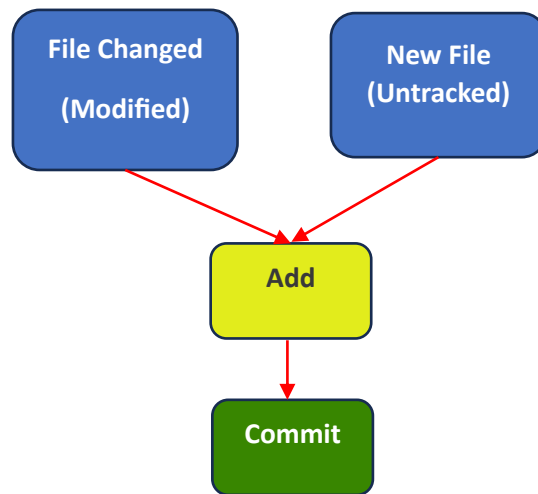
```
PS C:\Git and GitHub\Git_Demo\Project-Demo> git status
On branch main
Your branch is up to date with 'origin/main'.

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:   README.md

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        index.html

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

- Various update modes:
 - Untracked – new files that git doesn't yet track
 - Modified – changed
 - Staged – file is ready to be committed
 - Unmodified – unchanged
- Making changes permanent is a two-step process → add + commit
- If we add the updated or new created files it goes into the staged mode. Means files are ready for commit.



- Let's explore Add and Commit command:

Add & Commit

add - adds new or changed files in your working directory to the Git staging area.

```
git add <- file name ->
```

commit - it is the record of change

```
git commit -m "some message"
```

- Add below code in index.html file:

```
1 <h1>Hello World...!</h1>
```

- In the terminal type below command and check the status:

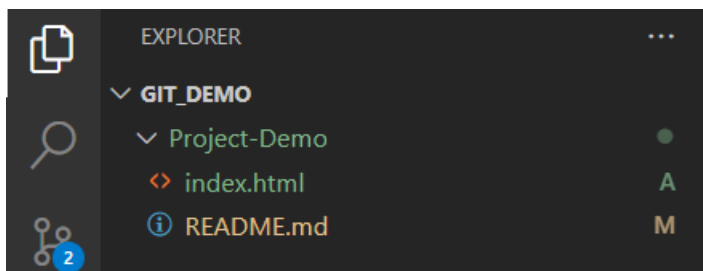


```
PS C:\Git and GitHub\Git_Demo\Project-Demo> git add index.html
● PS C:\Git and GitHub\Git_Demo\Project-Demo> git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   index.html

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:   README.md
```

- You will see the status changed and it shows staged. Also check explorer it shows A symbol next to index.html file which means file is Added.



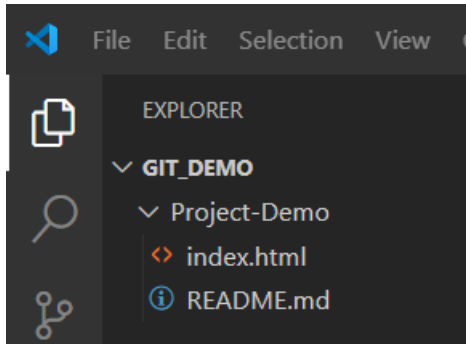
- To add Readme file we can write → git add README.md or we can simply write git add . command (where . indicates all the files) as mentioned below:

```
● PS C:\Git and GitHub\Git_Demo\Project-Demo> git add .
● PS C:\Git and GitHub\Git_Demo\Project-Demo> git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md
        new file:   index.html
```

- Let's commit the files in the GitHub. For committing the files execute below command:

```
● PS C:\Git and GitHub\Git_Demo\Project-Demo> git commit -m "Add new element"
[main 4db54d2] Add new element
2 files changed, 2 insertions(+), 1 deletion(-)
create mode 100644 index.html
```



- Check the git status:

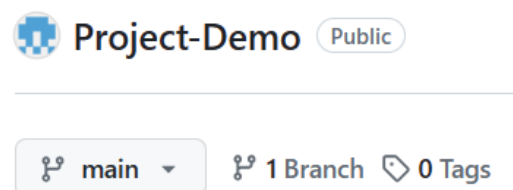
```
PS C:\Git and GitHub\Git_Demo\Project-Demo> git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

- You will see that the changes made in the project is still not visible in the GitHub. Check the message in the terminal:

Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

- We will use Push command to upload local repo content to remote repo as shown below:

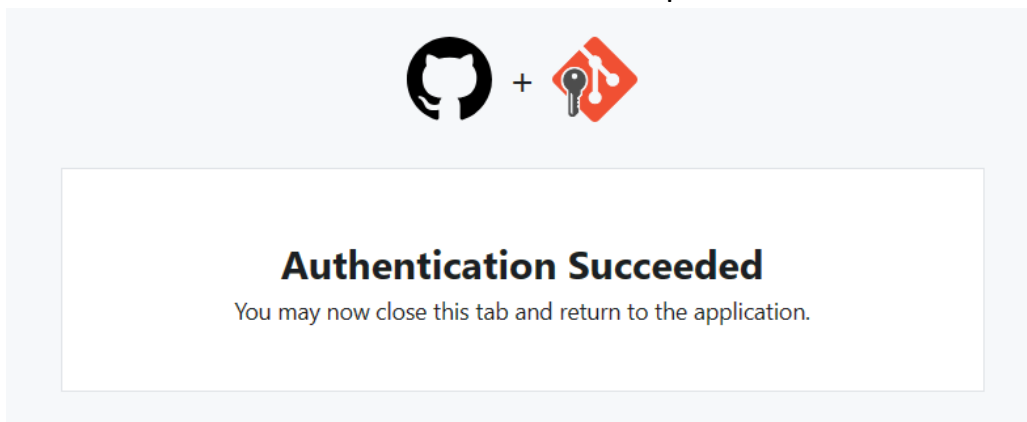


origin → Default folder (Project-Demo)
main → branch

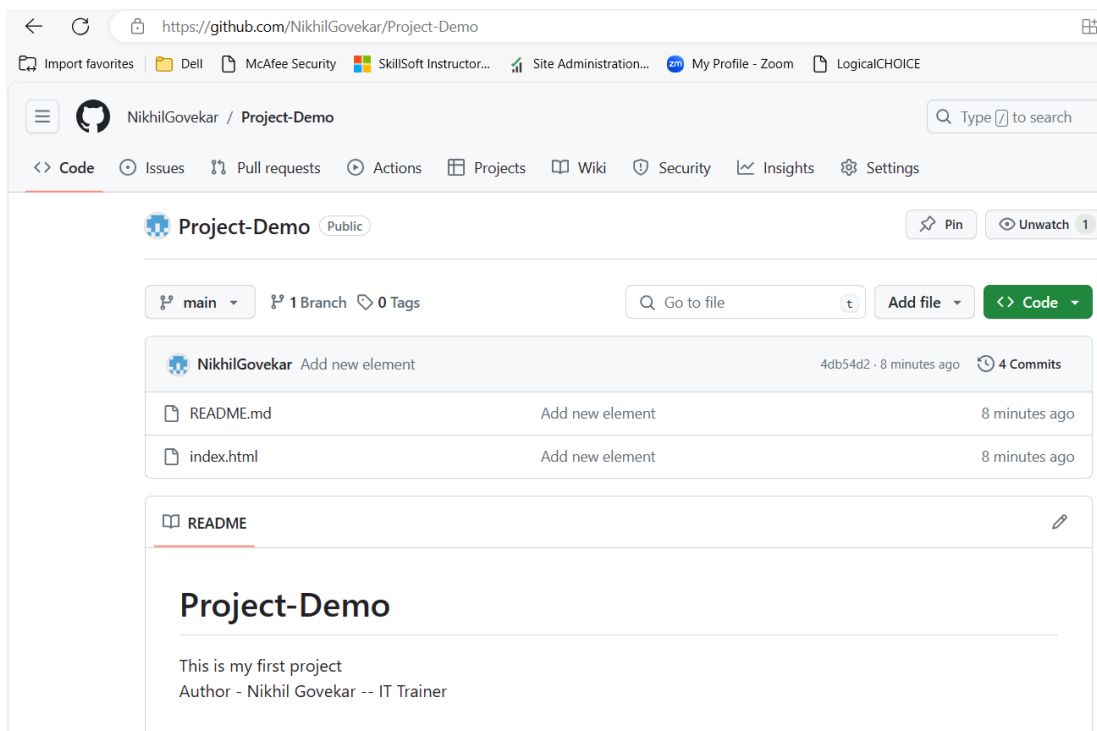


```
PS C:\Git and GitHub\Git_Demo\Project-Demo> git push origin main
info: please complete authentication in your browser...
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 360 bytes | 360.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/NikhilGovekar/Project-Demo.git
13b1028..4db54d2  main -> main
```

Note: After we give push command we need to authorize with GitHub. Once the authentication is successful it will push the files to GitHub.



- Check GitHub Repo and you will see that the changes are reflected:





- You can click on commits option on GitHub to view all the commits done till now:

Commits

main	All users	All time
Commits on Mar 7, 2024		
Add new element NikhilGovekar committed 10 minutes ago	4db54d2	<>
Added Next Line NikhilGovekar committed 3 hours ago	Verified 13b1028	<>
Update README.md NikhilGovekar committed 3 hours ago	Verified e1c649c	<>
Initial commit NikhilGovekar committed 4 hours ago	Verified 7a74d3e	<>

In the previous scenario we created Repo on GitHub and then we got it in the Local Machine as local Repo.

What is if create a Project on local machine first and then we want to push the project on GitHub.

We will need Init commands for the same:

Init Command

init - used to create a new git repo

`git init`

`git remote add origin <- link ->`

`git remote -v` (to verify remote)

`git branch` (to check branch)

`git branch -M main` (to rename branch)

`git push origin main`



Let's under the concept practically:

- Create a folder LocalRepo in GIT_DEMO Folder and traverse to LocalRepo folder as shown below:

```
PS C:\Git and GitHub\Git_Demo\Project-Demo> cd..
PS C:\Git and GitHub\Git_Demo> mkdir LocalRepo

Directory: C:\Git and GitHub\Git_Demo

Mode                LastWriteTime         Length Name
----                -
d-----          07-03-2024    15:56             LocalRepo

PS C:\Git and GitHub\Git_Demo> cd .\LocalRepo\
PS C:\Git and GitHub\Git_Demo\LocalRepo> 
```

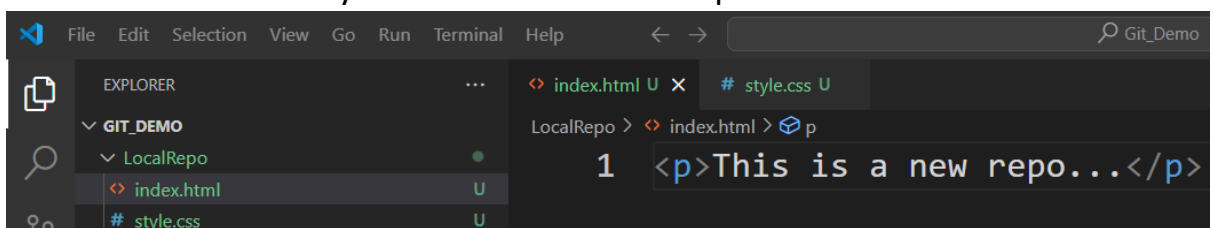
- Currently LocalRepo is a normal folder. We want to make this folder a git repo. In order to do that we need to execute git init command as shown below:

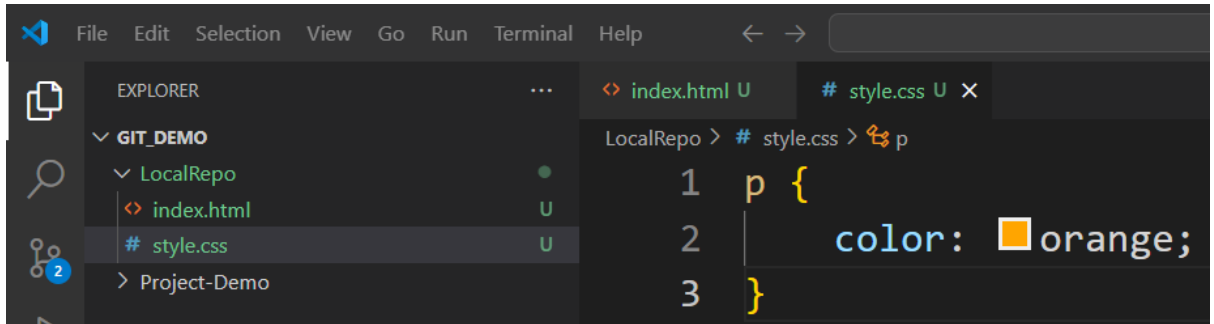
```
PS C:\Git and GitHub\Git_Demo\LocalRepo> Get-ChildItem -Force
PS C:\Git and GitHub\Git_Demo\LocalRepo> git init
Initialized empty Git repository in C:/Git and GitHub/Git_Demo/LocalRepo/.git/
PS C:\Git and GitHub\Git_Demo\LocalRepo> Get-ChildItem -Force

Directory: C:\Git and GitHub\Git_Demo\LocalRepo

Mode                LastWriteTime         Length Name
----                -
d--h--          07-03-2024    16:00             .git
```

- We can now see “.git” folder under LocalRepo which indicates it's a git repo.
- Add index.html and style.css file under LocalRepo folder as shown below:





- We can see that we have untracked files in the LocalRepo folder.

```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        index.html
        style.css

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

- Let's add all the files and check the status as shown below:

```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git add .
PS C:\Git and GitHub\Git_Demo\LocalRepo> git status
On branch main

No commits yet

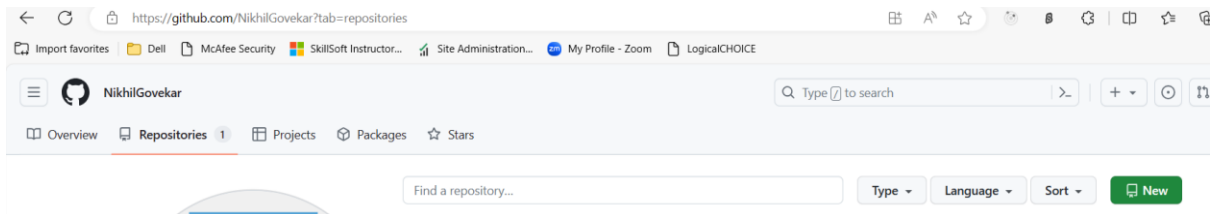
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   index.html
        new file:   style.css
```

- Let's commit the files and check the status as shown below:

```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git commit -m "Add Initial Files"
[main (root-commit) 6e0066a] Add Initial Files
 2 files changed, 4 insertions(+)
 create mode 100644 index.html
 create mode 100644 style.css
PS C:\Git and GitHub\Git_Demo\LocalRepo> git status
On branch main
nothing to commit, working tree clean
```



- Now we want to upload this Project – LocalRepo on the GitHub. In order to complete the process, follow the below mentioned steps:
- Go to GitHub → Your Profile → Repositories → Click on New button to create a new Repository.




- Fill in the details as shown below:

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).



Owner * Repository name *

 NikhilGovekar /

✓ LocalRepo is available.

Great repository names are short and memorable. Need inspiration? How about [musical-waddle](#) ?

Description (optional)

- ☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.
- ☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

- ☐ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)


*** Do not select README file for now.**

- Click on Create Repository Button. Repository LocalRepo will be created.



- Some instructions will be given by GitHub. We can keep this for our reference:

Quick setup — if you've done this kind of thing before

 Set up in Desktop or [HTTPS](#) [SSH](#) <https://github.com/NikhilGovekar/LocalRepo.git>

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# LocalRepo" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/NikhilGovekar/LocalRepo.git
git push -u origin main
```



...or push an existing repository from the command line

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

- Now before pushing the files from Machine LocalRepo to GitHub LocalRepo we need to execute below command:

`git remote add origin <<link>>` → Link you will get from GitHub

Quick setup — if you've done this kind of thing before

 Set up in Desktop or [HTTPS](#) [SSH](#) <https://github.com/NikhilGovekar/LocalRepo.git> 

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

<https://github.com/NikhilGovekar/LocalRepo.git>

This will set our origin – LocalRepo

```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git remote add origin https://github.com/NikhilGovekar/LocalRepo.git
```

- To check what origin is set, execute below command:

```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git remote -v
origin https://github.com/NikhilGovekar/LocalRepo.git (fetch)
origin https://github.com/NikhilGovekar/LocalRepo.git (push)
```




This will verify remote. We can see LocalRepo.git mentioned

- To check the branch under LocalRepo we can execute below command:

```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git branch
* main
```

- To push the project under main branch of GitHub LocalRepo, execute the below command:

`git push origin main`

OR

`git push -u origin main`

Note: -u creates an upstream in a simple word's shortcut for origin main branch. So in future we can simply write "git push" instead "git push origin main".

```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git push -u origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 317 bytes | 317.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/NikhilGovekar/LocalRepo.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

- Check the GitHub LocalRepo. You will see all the files uploaded on GitHub LocalRepo folder now.

LocalRepo Public

main 1 Branch 0 Tags

Go to file Add file Code

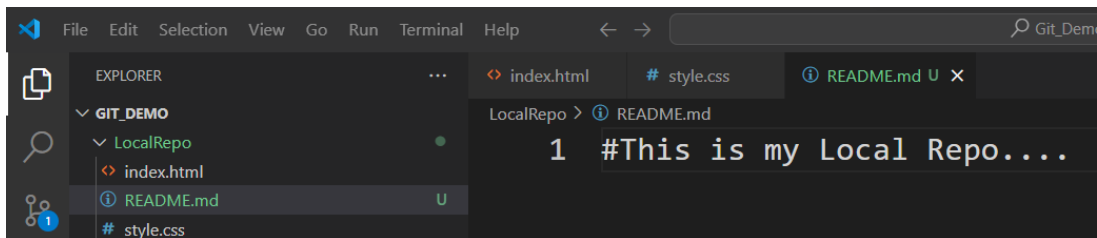
NikhilGovekar Add Initial Files 6e0066a · 29 minutes ago 1 Commits

index.html	Add Initial Files	29 minutes ago
style.css	Add Initial Files	29 minutes ago

Nikhil A. Govekar
Contact: +91 9819425121
Email ID: Nikhil.govekar@gmail.com



- Create README.md file in your machine LocalRepo folder as shown below:



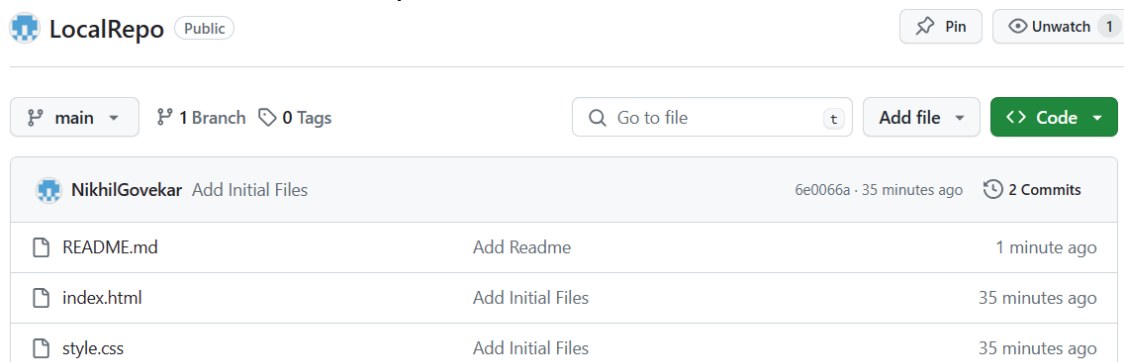
- Check the git status ... Then add and commit and to push the file to GitHub LocalRepo folder simple write git push as shown below:

```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        README.md

nothing added to commit but untracked files present (use "git add" to track)
PS C:\Git and GitHub\Git_Demo\LocalRepo> git add .
PS C:\Git and GitHub\Git_Demo\LocalRepo> git commit -m "Add Readme"
[main 05499d8] Add Readme
1 file changed, 1 insertion(+)
create mode 100644 README.md
PS C:\Git and GitHub\Git_Demo\LocalRepo> git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 16 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 (from 0)
To https://github.com/NikhilGovekar/LocalRepo.git
6e0066a..05499d8 main -> main
```

- Check the GitHub Local Repo folder:





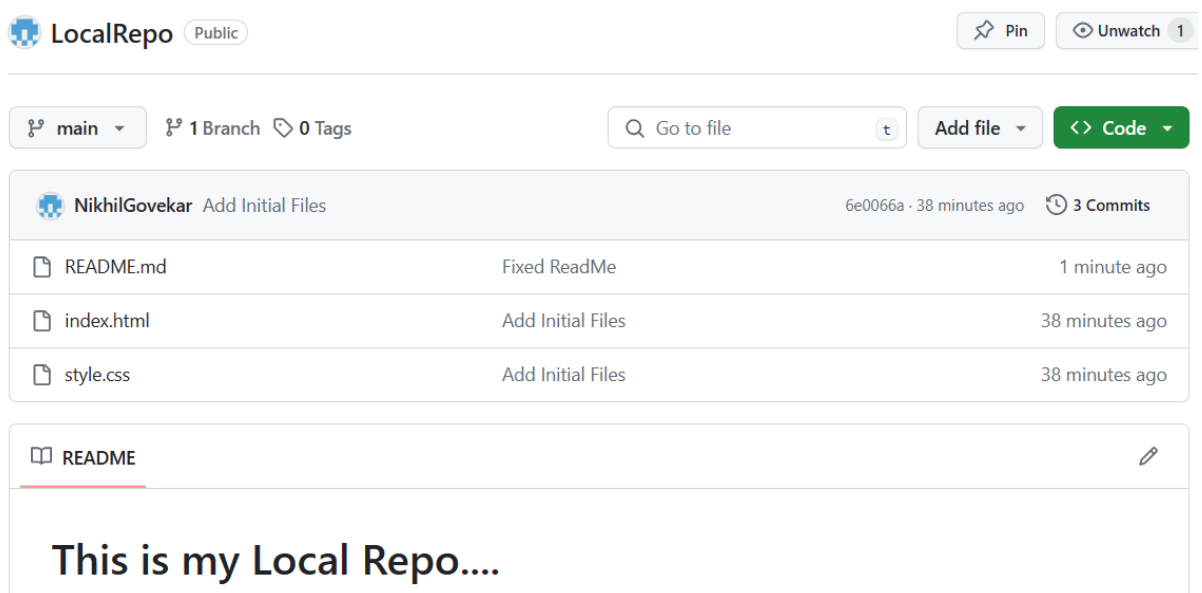
- Let's fix README file as shown below:



- Execute below command to add, commit and push file to GitHub LocalRepo folder:

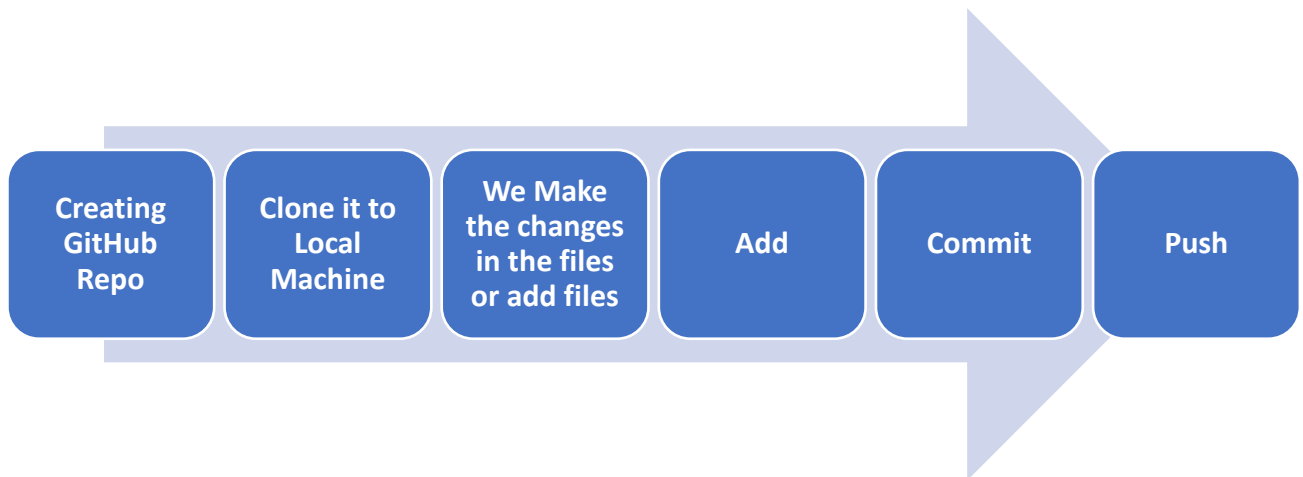
```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git add .
PS C:\Git and GitHub\Git_Demo\LocalRepo> git commit -m "Fixed ReadMe"
[main 4e646c4] Fixed ReadMe
 1 file changed, 1 insertion(+), 1 deletion(-)
PS C:\Git and GitHub\Git_Demo\LocalRepo> git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 341 bytes | 341.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/NikhilGovekar/LocalRepo.git
05499d8..4e646c4  main -> main
```

- View the changes in GitHub LocalRepo:

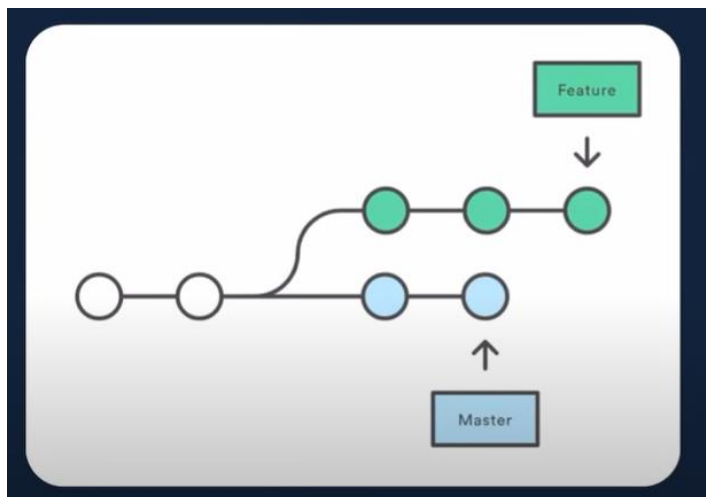




General Workflow followed in the industry:



Git Branches:



Note: Master is a main branch

Branching in Git is a helpful feature for software developers working on a big team project. It allows the team members to work on different aspects of the software by creating a branch from the main branch. The main branch is not affected by the changes in the created branch until it is merged into the main branch. Once, work is complete for the branch, it can be merged into the main branch to incorporate the changes. It can be used for better management of files related to the software. It also helps us try or experiment with adding some new features to the code; if it works well, it can be merged with the main code in the main branch.



Branch Commands

`git branch` (to check branch)

`git branch -M main` (to rename branch)

`git checkout <- branch name ->` (to navigate)

`git checkout -b <- new branch name ->` (to create new branch)

`git branch -d <- branch name ->` (to delete branch)

- Go to terminal and type below command to check the current branch name:

```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git branch
* main
```

- To create a new branch “feature1” type below command:

```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git checkout -b feature1
Switched to a new branch 'feature1'
PS C:\Git and GitHub\Git_Demo\LocalRepo> git branch
* feature1
  main
```

- We can see that currently we are at feature1 branch... current branch is highlighted in **green**
- To go to “main” branch, we can write below command:



```
● PS C:\Git and GitHub\Git_Demo\LocalRepo> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
● PS C:\Git and GitHub\Git_Demo\LocalRepo> git branch
feature1
* main
```

- This time main is highlighted in green since main is the current branch we switched to.
- Let's create one more branch "feature2" and check the current branch:

```
● PS C:\Git and GitHub\Git_Demo\LocalRepo> git checkout -b feature2
Switched to a new branch 'feature2'
● PS C:\Git and GitHub\Git_Demo\LocalRepo> git branch
feature1
* feature2
main
```

- To delete the branch, type the below command:

Note: We cannot delete the branch if we are already on the same branch. See the below error:

```
⊗ PS C:\Git and GitHub\Git_Demo\LocalRepo> git branch -d feature2
error: cannot delete branch 'feature2' used by worktree at 'C:/Git and GitHub/Git_Demo/LocalRepo'
```

- Switch to another branch and then delete "feature2" branch as shown below:

```
● PS C:\Git and GitHub\Git_Demo\LocalRepo> git checkout main
● Switched to branch 'main'
Your branch is up to date with 'origin/main'.
● PS C:\Git and GitHub\Git_Demo\LocalRepo> git branch
feature1
feature2
* main
● PS C:\Git and GitHub\Git_Demo\LocalRepo> git branch -d feature2
Deleted branch feature2 (was 4e646c4).
● PS C:\Git and GitHub\Git_Demo\LocalRepo> git branch
feature1
* main
```

- Let's add new feature in "feature1" branch. For that go to "feature1" branch first:



```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git checkout feature1
Switched to branch 'feature1'
PS C:\Git and GitHub\Git_Demo\LocalRepo> git branch
* feature1
  main
```

- Now to index.html and add the new paragraph tag as shown below:

```
<> index.html M x # style.css ⓘ README.md
LocalRepo > <> index.html > p
1 <p>This is a new repo...</p>
2 <p>New feature Added</p>
```

- Check the status:

```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git status
On branch feature1
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:   index.html

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

Under “feature1” branch modification is done in index.html.

- Add and commit the changes as shown below:

```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git add .
PS C:\Git and GitHub\Git_Demo\LocalRepo> git commit -m "Add New Feature"
[feature1 d961acd] Add New Feature
 1 file changed, 2 insertions(+), 1 deletion(-)
PS C:\Git and GitHub\Git_Demo\LocalRepo> git status
On branch feature1
nothing to commit, working tree clean
```

- We can see no more further changes to commit in branch “feature1”.
- Now checkout to “main” branch:



```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

- You will see that the new changes will not be seen in index.html:

```
<> index.html X # style.css ⓘ README.md
LocalRepo > <> index.html > p
1 <p>This is a new repo...</p>
```

- The new changes done is available only in “feature1” branch. Switch to feature1 branch and check index.html file:

```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git checkout feature1
Switched to branch 'feature1'
```

```
<> index.html X # style.css ⓘ README.md
LocalRepo > <> index.html > p
1 <p>This is a new repo...</p>
2 <p>New feature Added</p>
```

- Now we need to push these changes in GitHub repository – LocalRepo. To push write the command - git push origin feature1

```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git branch
* feature1
  main
PS C:\Git and GitHub\Git_Demo\LocalRepo> git push origin feature1
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 367 bytes | 367.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'feature1' on GitHub by visiting:
remote:   https://github.com/NikhilGovekar/LocalRepo/pull/new/feature1
remote:
To https://github.com/NikhilGovekar/LocalRepo.git
 * [new branch]   feature1 -> feature1
```




- Go to GitHub and check the new branch “feature1” will be seen:

https://github.com/NikhilGovekar/LocalRepo

Import favorites | Dell | McAfee Security | SkillSoft Instructor... | Site Administration... | My Profile - Zoom | LogicalCHOICE

NikhilGovekar / LocalRepo

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

LocalRepo Public

feature1 had recent pushes 1 minute ago

Compare & pull request

main 2 Branches 0 Tags

Go to file Add file Code

Switch branches/tags

Find or create a branch...

Branches Tags

main default

feature1

View all branches

4e646c4 · 5 hours ago 3 Commits

Fixed ReadMe 5 hours ago

Add Initial Files 6 hours ago

Add Initial Files 6 hours ago

This is my Local Repo....

- Click on feature1 and check the files:



feature1 2 Branches 0 Tags

Go to file Add file Code

This branch is 1 commit ahead of main.

Contribute

NikhilGovekar Add New Feature d961acd · 36 minutes ago 4 Commits

README.md	Fixed ReadMe	5 hours ago
index.html	Add New Feature	36 minutes ago
style.css	Add Initial Files	6 hours ago

README

This is my Local Repo....

- Click on index.html and check the output:

NikhilGovekar / LocalRepo

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Files

feature1

Go to file

README.md index.html style.css

LocalRepo / index.html

NikhilGovekar Add New Feature

Code Blame 2 lines (2 loc) · 53 Bytes Code 55% faster with GitHub Copilot

```
1 <p>This is a new repo...</p>
2 <p>New feature Added</p>
```

- Now we want to merge both the branches – main branch and feature1 branch.
- For merging branches there are two approaches:



Merging Code

Way 1

`git diff <- branch name->` (to compare commits, branches, files & more)

`git merge <- branch name->` (to merge 2 branches)

Way 2

Create a PR

PR stands for Pull Request

- Let's explore both the approaches one by one.
- First step is to compare differences between both the branches (feature1 with main branch). Write the below command for the same:

```
● PS C:\Git and GitHub\Git_Demo\LocalRepo> git branch
* feature1
  main
● PS C:\Git and GitHub\Git_Demo\LocalRepo> git diff main
diff --git a/index.html b/index.html
index b9905ab..38b6d1a 100644
--- a/index.html
+++ b/index.html
@@ -1,2 @@
-<p>This is a new repo...</p>
\ No newline at end of file
+<p>This is a new repo...</p>
+<p>New feature Added</p>
\ No newline at end of file
```

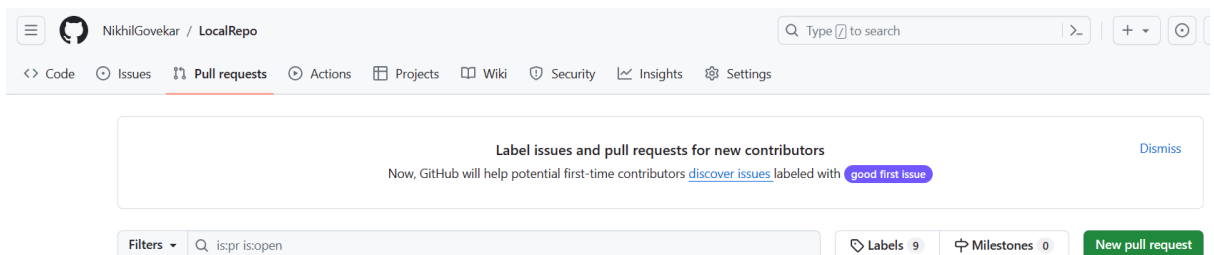
- Now let's create Pull Request (PR) to merge the branches.

Pull Request

It lets you tell others about changes you've pushed to a branch in a repository on GitHub.



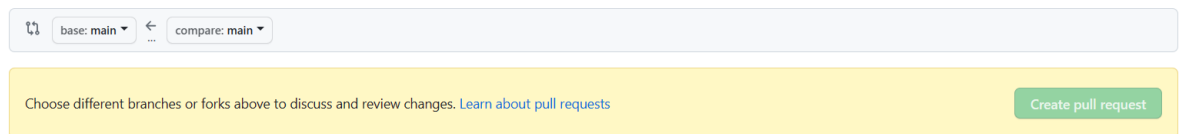
- Once the PR is created the senior developer or manager reviews the PR and give the comment whether to merge the branches or not.
- In our case we will create PR and merge the branches as shown below.
- Go to GitHub LocalRepo – feature1, you see a button Compare and Pull Request.
- If not then go the pull requests tab → Click on New Pull Request button



- It will ask to compare changes between branches as mentioned below:

Compare changes

Compare changes across branches, commits, tags, and more below. If you need to, you can also [compare across forks](#).



Compare and review just about anything

Branches, tags, commit ranges, and time ranges. In the same repository and across forks.

Example comparisons	
feature1	13 hours ago
main@{1day}...main	24 hours ago

- Click on feature1 → Comparing changes will be seen:



Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#) or [learn more about diff comparisons](#).

base: main ← compare: feature1 ✓ **Able to merge.** These branches can be automatically merged.

Discuss and review the changes in this comparison with others. [Learn about pull requests](#) [Create pull request](#)

1 commit 1 file changed 1 contributor

Commits on Mar 7, 2024

Add New Feature

NikhilGovekar committed 14 hours ago

d961acd

Showing 1 changed file with 2 additions and 1 deletion. Split Unified

3 index.html

@@ -1,2 @@

1 - <p>This is a new repo...</p>

1 + <p>This is a new repo...</p>

2 + <p>New feature Added</p>

- Click on Create pull request button.

Open a pull request

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

base: main ← compare: feature1 ✓ **Able to merge.** These branches can be automatically merged.

Add a title

Add New Feature

Add a description

Write Preview

H B I

Add your description here...

Markdown is supported

Paste, drop, or click to add files

Reviewers

No reviews

Assignees

No one—[assign yourself](#)

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

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



Helpful resources



Create pull request

- Click on Create pull request. GitHub will check what automatically can be merged. Click on Merge pull request button.



Add New Feature #1

 Open NikhilGovekar wants to merge 1 commit into `main` from `feature1` 

 Conversation **0**  Commits **1**  Checks **0**  Files changed **1**




NikhilGovekar commented 1 minute ago

Owner ...

No description provided.




 Add New Feature

d961acd

Add more commits by pushing to the `feature1` branch on [NikhilGovekar/LocalRepo](#).



 **Require approval from specific reviewers before merging**

Add rule ×

[Rulesets](#) ensure specific people approve pull requests before they're merged.



Continuous integration has not been set up

[GitHub Actions](#) and [several other apps](#) can be used to automatically catch bugs and enforce style.



This branch has no conflicts with the base branch

Merging can be performed automatically.

Merge pull request

You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

- Click on Confirm Merge. You will see the new commit is added with the name “Merge pull request #1 from NikhilGovekar/feature1”.



Add New Feature #1

Open NikhilGovekar wants to merge 1 commit into `main` from `feature1`

Conversation 0 Commits 1 Checks 0 Files changed 1

NikhilGovekar commented 3 minutes ago Owner ...

No description provided.

Add New Feature d961acd

Add more commits by pushing to the `feature1` branch on [NikhilGovekar/LocalRepo](#).

Merge pull request #1 from NikhilGovekar/feature1

Add New Feature

This commit will be authored by 162542401+NikhilGovekar@users.noreply.github.com

Confirm merge Cancel

- You will see merge pull request is successfully merged and closed.

Add New Feature #1

Merged NikhilGovekar merged 1 commit into `main` from `feature1` now

Conversation 0 Commits 1 Checks 0 Files changed 1

NikhilGovekar commented 3 minutes ago Owner ...

No description provided.

Add New Feature d961acd

NikhilGovekar merged commit `f496e01` into `main` now Revert

Pull request successfully merged and closed Delete branch

You're all set—the `feature1` branch can be safely deleted.

- Our pull request is not closed. Go to the code and review the `index.html` file under `main` branch. You will see new data merged in it.



LocalRepo Public

Pin

Unwatch 1

main 2 Branches 0 Tags

Go to file

Add file

Code

NikhilGovekar	Merge pull request #1 from NikhilGovekar/feature1	f496e01 · 2 minutes ago	5 Commits
README.md	Fixed ReadMe	19 hours ago	
index.html	Add New Feature	14 hours ago	
style.css	Add Initial Files	19 hours ago	

NikhilGovekar / LocalRepo

<> Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Files

main

Go to file

README.md

index.html

style.css

LocalRepo / index.html

NikhilGovekar Add New Feature

Code Blame 2 lines (2 loc) · 53 Bytes

1 <p>This is a new repo...</p>

2 <p>New feature Added</p>

- Now go to your LocalRepo on machine and checkout on main branch.

```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Git and GitHub\Git_Demo\LocalRepo> git branch
feature1
* main
```

- Check index.html file – you will find that the second paragraph is still not seen in the file.

```
index.html x # style.css i README.md
LocalRepo > index.html > p
1 <p>This is a new repo...</p>
```




- Feature is merged on GitHub. It is still not available in the Machine LocalRepo.
- When we want changes from remote to local i.e. GitHub LocalRepo to Machine LocalRepo we use pull command.

Pull Command

```
git pull origin main
```

used to fetch and download content from a remote repo and immediately update the local repo to match that content.

- Execute pull command as shown below:

```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git pull origin main
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (1/1), 900 bytes | 450.00 KiB/s, done.
From https://github.com/NikhilGovekar/LocalRepo
* branch      main      -> FETCH_HEAD
   4e646c4..f496e01  main      -> origin/main
Updating 4e646c4..f496e01
Fast-forward
 index.html | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)
```

- You will the changes is available in the index.html file now:

```
<> index.html X # style.css ⓘ README.md
LocalRepo > <> index.html > p
1  <p>This is a new repo...</p>
2  <p>New feature Added</p>
```

- Let's explore how to handle merge conflicts in Git.



Resolving Merge Conflicts

An event that takes place when Git is unable to automatically resolve differences in code between two commits.

- Conflict arises when the changes are found in the same line of code in the file across branches and git is not able to resolve it automatically. Then we need to resolve these changes manually.
- In order to do so let's make changes in index.html file in main branch:

```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git branch
feature1
* main
```

```
<> index.html X # style.css ⓘ README.md
LocalRepo > <> index.html > p
1 <p>This is a new repo...</p>
2 <p>New feature Added (button)</p>
```

- Add and commit the file as shown below:

```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git add .
PS C:\Git and GitHub\Git_Demo\LocalRepo> git commit -m "Add Button"
[main 63143f1] Add Button
1 file changed, 1 insertion(+), 1 deletion(-)
```

- Now checkout to feature1 and add the below details in index.html file:

```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git checkout feature1
Switched to branch 'feature1'

<> index.html M X # style.css ⓘ README.md
LocalRepo > <> index.html > p
1 <p>This is a new repo...</p>
2 <p>New feature Added (dropdown)</p>
```



- As we are currently in feature1 branch, add and commit the file as shown below:

```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git branch
* feature1
  main
PS C:\Git and GitHub\Git_Demo\LocalRepo> git add .
PS C:\Git and GitHub\Git_Demo\LocalRepo> git commit -m "Add Dropdown"
[feature1 b0d2e46] Add Dropdown
1 file changed, 1 insertion(+), 1 deletion(-)
```

- Now we have different changes in main and feature1 branches. Let's check the differences with main as currently we are in feature1 branch:

```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git branch
* feature1
  main
PS C:\Git and GitHub\Git_Demo\LocalRepo> git diff main
diff --git a/index.html b/index.html
index 0843299..2f8ca22 100644
--- a/index.html
+++ b/index.html
@@ -1,2 +1,2 @@
 <p>This is a new repo...</p>
-<p>New feature Added (button)</p>
\ No newline at end of file
+<p>New feature Added (dropdown)</p>
\ No newline at end of file
```

- Now let's merge both the branches. We have previously done merging with the help of Pull Request (PR), but this time we will use second approach – git merge command:

```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git merge main
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.
```

- This shown the conflict between the files. It is an event which takes place when Git is unable to automatically resolve difference between the two committed code:



```
1 <p>This is a new repo...</p>
   Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
2 <<<<<< HEAD (Current Change)
3 <p>New feature Added (dropdown)</p>
4 =====
5 <p>New feature Added (button)</p>
6 >>>>> main (Incoming Change)
7
```

- Now we can see that git is not able to resolve differences and results into conflict. In this case we have to manually resolve this.
 - Accept Current Change – Accept feature1 change
 - Accept Incoming Change – Accept change coming from main branch
 - Accept Both Changes – Accept changes from both files
- To resolve it manually – remove all the additional code seen in index.html file:

1. Now if we save file as mentioned below means we want both the changes:

```
1 <p>This is a new repo...</p>
2 <p>New feature Added (dropdown)</p>
3 <p>New feature Added (button)</p>
4
```

2. If we save the file as mentioned below means we want change from feature1 branch:



```
<> index.html ! x # style.css ⓘ README.md
LocalRepo > <> index.html > ...
1 <p>This is a new repo...</p>
2 <p>New feature Added (dropdown)</p>
3
```

3. If we save the file as mentioned below means we want change from main branch:

```
<> index.html ! x # style.css ⓘ README.md
LocalRepo > <> index.html > ...
1 <p>This is a new repo...</p>
2 <p>New feature Added (button)</p>
3
```

- We will keep changes from both the branches and check the status:

```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git status
On branch feature1
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:   index.html

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

- It shows we have modified file. Now add and commit the changes in the file as shown below:

```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git add .
PS C:\Git and GitHub\Git_Demo\LocalRepo> git commit -m "Add both features"
[feature1 33010a0] Add both features
```

- Check the status again. Ensure there is nothing to commit.



```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git status
On branch feature1
nothing to commit, working tree clean
```

- Check difference with main now:

```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git diff main
diff --git a/index.html b/index.html
index 0843299..dbcb883 100644
--- a/index.html
+++ b/index.html
@@ -1,2 +1,4 @@
 <p>This is a new repo...</p>
-<p>New feature Added (button)</p>
\ No newline at end of file
+<p>New feature Added (dropdown)</p>
+<p>New feature Added (button)</p>
+
```

We have one change more.

- Now checkout to main branch and check the index.html file. You not find the feature in it:

```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)
```

```
<> index.html X # style.css ⓘ README.md
LocalRepo > <> index.html > p
1 <p>This is a new repo...</p>
2 <p>New feature Added (button)</p>
```

- Now we need to merge feature1 branch details with main. Follow the steps mentioned below:



```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git branch
feature1
* main
● PS C:\Git and GitHub\Git_Demo\LocalRepo> git merge feature1
Updating 63143f1..33010a0
Fast-forward
 index.html | 4 +++-
 1 file changed, 3 insertions(+), 1 deletion(-)
```

- You will see the details from feature1 is merged in main branch:

```
< index.html X # style.css ⓘ README.md
LocalRepo > < index.html > ...
1 <p>This is a new repo...</p>
2 <p>New feature Added (dropdown)</p>
3 <p>New feature Added (button)</p>
```

- Now push these changes to GitHub LocalRepo and check the output in GitHub:

```
● PS C:\Git and GitHub\Git_Demo\LocalRepo> git push
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 16 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (9/9), 1.02 KiB | 523.00 KiB/s, done.
Total 9 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/NikhilGovekar/LocalRepo.git
 f496e01..33010a0 main -> main
```



NikhilGovekar / LocalRepo

Public

main 2 Branches 0 Tags

Go to file Add file Code

NikhilGovekar Add both features 33010a0 · 12 minutes ago 8 Commits

README.md	Fixed ReadMe	20 hours ago
index.html	Add both features	12 minutes ago
style.css	Add Initial Files	yesterday

README

This is my Local Repo....

Open index.html file under main branch and check.

Let's understand undoing changes in Git:

Undoing Changes

Case 1 : staged changes

```
git reset <- file name ->
git reset
```

Case 2 : committed changes (for one commit)

```
git reset HEAD~1
```

Case 3 : committed changes (for many commits)

```
git reset <- commit hash ->
git reset --hard <- commit hash ->
```

Staged changes → Which are added but not committed.



- Let's explore undoing staged changes. Check the current branch and ensure we are at main branch:

```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git branch
feature1
* main
```

- Delete button code line from index.html file:

```
<> index.html M x # style.css i README.md
LocalRepo > <> index.html > ...
1 <p>This is a new repo...</p>
2 <p>New feature Added (dropdown)</p>
3
```

- Add the changes and check the status as mentioned below:

```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git add .
PS C:\Git and GitHub\Git_Demo\LocalRepo> git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   index.html
```

- We can see changes are staged and ready for commit. Now we want to revert back to the previous state. To do that we can write below command:

git reset index.html (perform reset in particular file)

OR

git reset (perform reset in all files)

```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git reset
Unstaged changes after reset:
M    index.html
```

- Check the status and you will see it will be back to not staged state:



```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git status
On branch main
Your branch is up to date with 'origin/main'.

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:   index.html

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

Committed changes → Changes done in file are added and committed (for one commit).

- Let's say we have deleted button code in index.html file of main branch. We are currently at the same status...
- Add and commit the file and check the git status as shown below:

```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git add .
PS C:\Git and GitHub\Git_Demo\LocalRepo> git commit -m "Delete button"
[main 5a1223b] Delete button
1 file changed, 2 deletions(-)
```

```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

- It shows branch is ahead of origin / main. This is because in GitHub LocalRepo → Index.html file we have button code. But in Machine LocalRepo → Index.html it not present.
- If we want to undo single commit, we use below command:

`git reset HEAD~1`

HEAD~1 indicates last commit.

The command means revert to previous commit done.



```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git reset HEAD~1
Unstaged changes after reset:
M      index.html
```

- All the changes are now un-staged. Check the status:

```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git status
On branch main
Your branch is up to date with 'origin/main'.

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:   index.html

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

- We can check all out commits done till now using command – git log

```
commit 33010a0f120e39e7f227006a91f35796cea95755 (HEAD -> main, origin/main, feature1)
Merge: b0d2e46 63143f1
Author: NikhilGovekar <nikhil.govekar@gmail.com>
Date:   Fri Mar 8 13:21:46 2024 +0530

    Add both features

commit b0d2e46414f2c369fcd794fce8de317ee4df47f
Author: NikhilGovekar <nikhil.govekar@gmail.com>
Date:   Fri Mar 8 12:47:04 2024 +0530

    Add Dropdown

commit 63143f1b9746967d1eb9f6dab4e6159698ab215e
Author: NikhilGovekar <nikhil.govekar@gmail.com>
Date:   Fri Mar 8 12:43:17 2024 +0530

    Add Button

commit f496e011adca4bd8eeeb647a2bc7238a38515fc7
Merge: 4e646c4 d961acd
Author: NikhilGovekar <162542401+NikhilGovekar@users.noreply.github.com>
:
```

- Currently it shows commits done under main branch with date and time and the HEAD position.



Committed changes → Reverting multiple commits performed in the branch

- For reverting multiple commits in the branch, we need to execute below command:

`git reset <commit hash>` → hash code is found under log (git log)

- Let's say want to go back to Add Button commit as mentioned below:

```
commit 33010a0f120e39e7f227006a91f35796cea95755 (HEAD -> main, origin/main, feature1)
Merge: b0d2e46 63143f1
Author: NikhilGovekar <nikhil.govekar@gmail.com>
Date: Fri Mar 8 13:21:46 2024 +0530

    Add both features

commit b0d2e46414f2c369fcd794fce8de317ee4df47f
Author: NikhilGovekar <nikhil.govekar@gmail.com>
Date: Fri Mar 8 12:47:04 2024 +0530

    Add Dropdown

commit 63143f1b9746967d1eb9f6dab4e6159698ab215e
Author: NikhilGovekar <nikhil.govekar@gmail.com>
Date: Fri Mar 8 12:43:17 2024 +0530

    Add Button

commit f496e011adca4bd8eeeb647a2bc7238a38515fc7
Merge: 4e646c4 d961acd
Author: NikhilGovekar <162542401+NikhilGovekar@users.noreply.github.com>
:
```

- Copy the hash code for the same and execute below command:

Hash code - 63143f1b9746967d1eb9f6dab4e6159698ab215e

```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git reset 63143f1b9746967d1eb9f6dab4e6159698ab215e
Unstaged changes after reset:
M      index.html
```

- Check git status



```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git status
On branch main
Your branch is behind 'origin/main' by 2 commits, and can be fast-forwarded.
(use "git pull" to update your local branch)

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:   index.html

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

- You will see we got all the changes back till the point mentioned (behind by 2 commits).
- We can check and log and find all commits are removed:

```
commit 63143f1b9746967d1eb9f6dab4e6159698ab215e (HEAD -> main)
Author: NikhilGovekar <nikhil.govekar@gmail.com>
Date:   Fri Mar 8 12:43:17 2024 +0530

    Add Button

commit f496e011adca4bd8eeeb647a2bc7238a38515fc7
Merge: 4e646c4 d961acd
Author: NikhilGovekar <162542401+NikhilGovekar@users.noreply.github.com>
Date:   Fri Mar 8 11:39:55 2024 +0530

    Merge pull request #1 from NikhilGovekar/feature1

    Add New Feature

commit d961acd879178aff0c64795903a5f8620225382d (origin/feature1)
Author: NikhilGovekar <nikhil.govekar@gmail.com>
Date:   Thu Mar 7 21:38:20 2024 +0530

    Add New Feature

commit 4e646c440dbf53f6e60502ab1d9925f77ae2cb28
:|
```

- The latest commit we are getting now is of Add Commit. Head is now changed.
- git reset removed changes from the git. But along with that if we want to remove changes from the code file as well, we need to use below command:

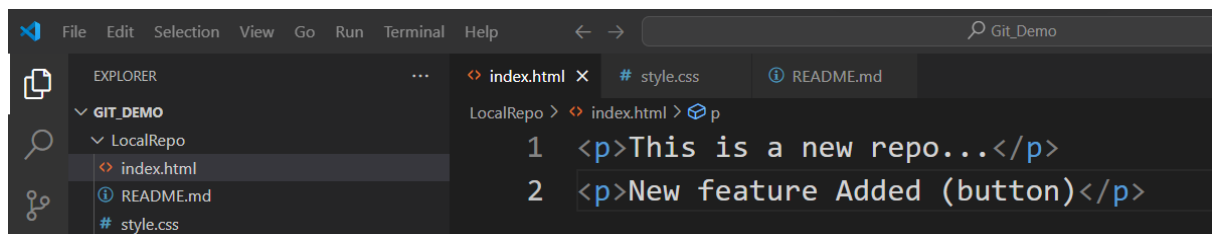


```
PS C:\Git and GitHub\Git_Demo\LocalRepo> git status
On branch main
Your branch is behind 'origin/main' by 2 commits, and can be fast-forwarded.
(use "git pull" to update your local branch)

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:   index.html

no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Git and GitHub\Git_Demo\LocalRepo> git reset --hard 63143f1b9746967d1eb9f6dab4e6159698ab215e
HEAD is now at 63143f1 Add Button
PS C:\Git and GitHub\Git_Demo\LocalRepo> git status
On branch main
Your branch is behind 'origin/main' by 2 commits, and can be fast-forwarded.
(use "git pull" to update your local branch)

nothing to commit, working tree clean
```



- We can see in the explorer no modified state... It is completely in the updated state.

Fork in Git:

Fork helps us in create a rough copy of the project:

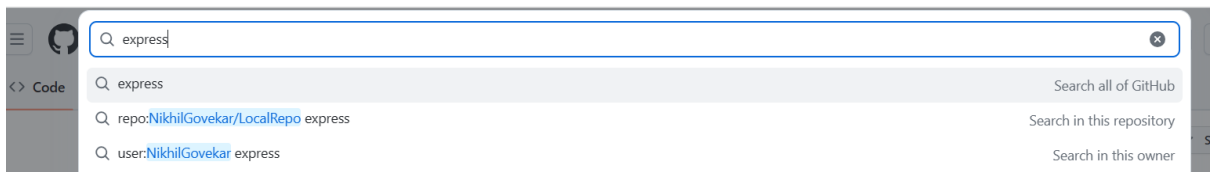
Fork

A fork is a new repository that shares code and visibility settings with the original “upstream” repository.

Fork is a rough copy.

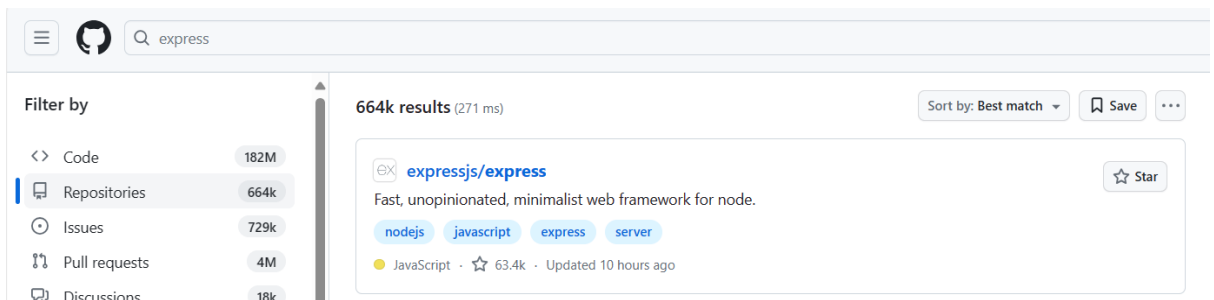
Let's say we want to make changes in some other developers project or contribute to any other open source project on GitHub.. This option is very useful.

- Search express project on GitHub:

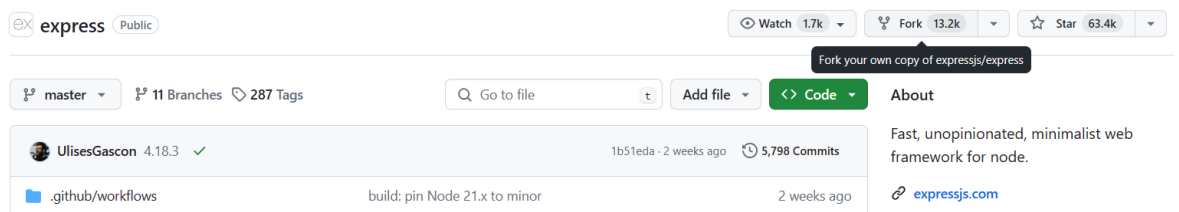


Click on Search all of GitHub.

- We will get the expressjs / express repo



- Click on the link expressjs/express and you can see the complete repo.
- Now we want to create the copy this express project in our account. In order to do that, we can fork the project as shown below.



- Click on Fork option and add below details:




Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. [View existing forks.](#)

Required fields are marked with an asterisk (*).

Owner * Repository name *

 NikhilGovekar /


✓ express is available.

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)

Fast, unopinionated, minimalist web framework for node.

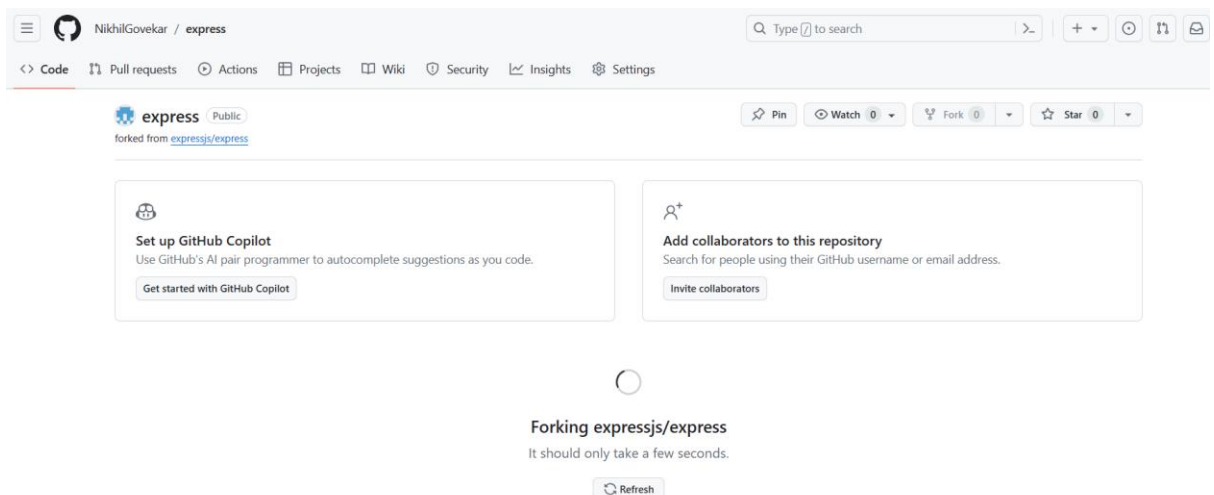
☐ Copy the **master** branch only
Contribute back to expressjs/express by adding your own branch. [Learn more.](#)

 You are creating a fork in your personal account.

Create fork

Uncheck the Copy the master branch only checkbox as we want complete project.

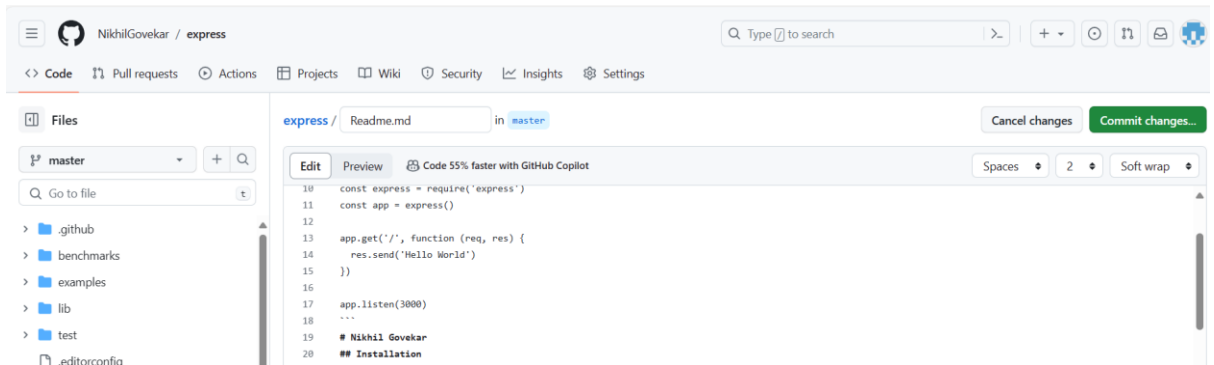
- Click on Create fork button. Forking started →



- Now we can see we have a complete express project in our GitHub Account. This how we can get the other project in our account to work the same.



- We can make the necessary changes in the project file. Open Readme file and write your name:



- Click on Commit Changes button.

Commit changes

Commit message

Update Readme.md

Extended description

Add an optional extended description..

☒ Commit directly to the master branch

☐ Create a new branch for this commit and start a pull request

[Learn more about pull requests](#)

Cancel

Commit changes

- Click on commit changes... This will commit in the express project of our account.



- We can see this branch is 1 commit ahead of expressjs/express master.
- To merge our changes in the original project we create Pull Request (PR)

- It shows what changes we have done... and click on the pull request to merge the changes....
- We must only create the pull request when we do some critical and necessary changes... in this case we will discard ...



So finally in this entire session we have created 3 repositories:

The screenshot shows the GitHub profile page for NikhilGovekar. The page includes a search bar, a list of repositories, and a profile section. The repositories listed are:

- express** (Public): Forked from expressjs/express. Fast, unopinionated, minimalist web framework for node. JavaScript, MIT License. Updated 5 minutes ago.
- LocalRepo** (Public): HTML. Updated 2 hours ago.
- Project-Demo** (Public): This is my first project. HTML. Updated yesterday.

The profile section shows the user's name, a bio, and a link to the profile page. The user joined yesterday.