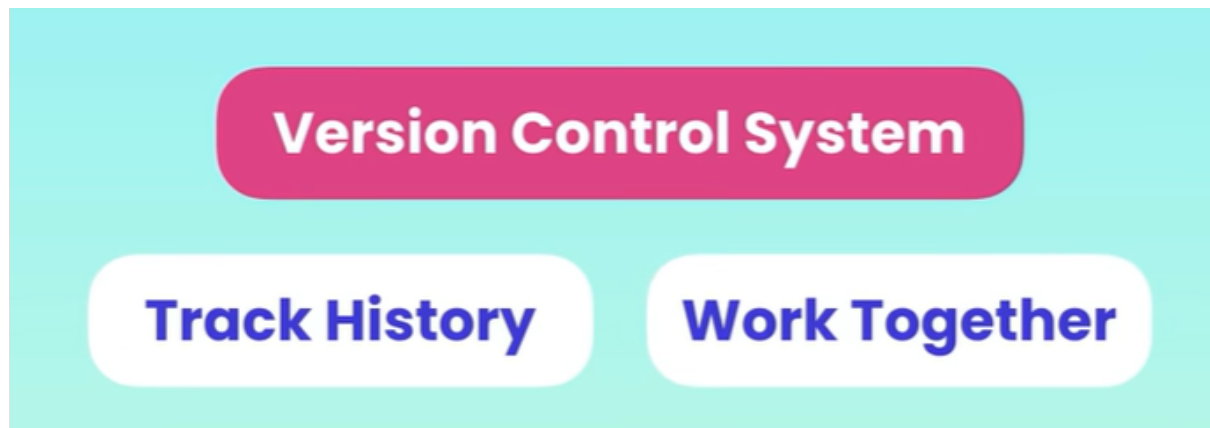


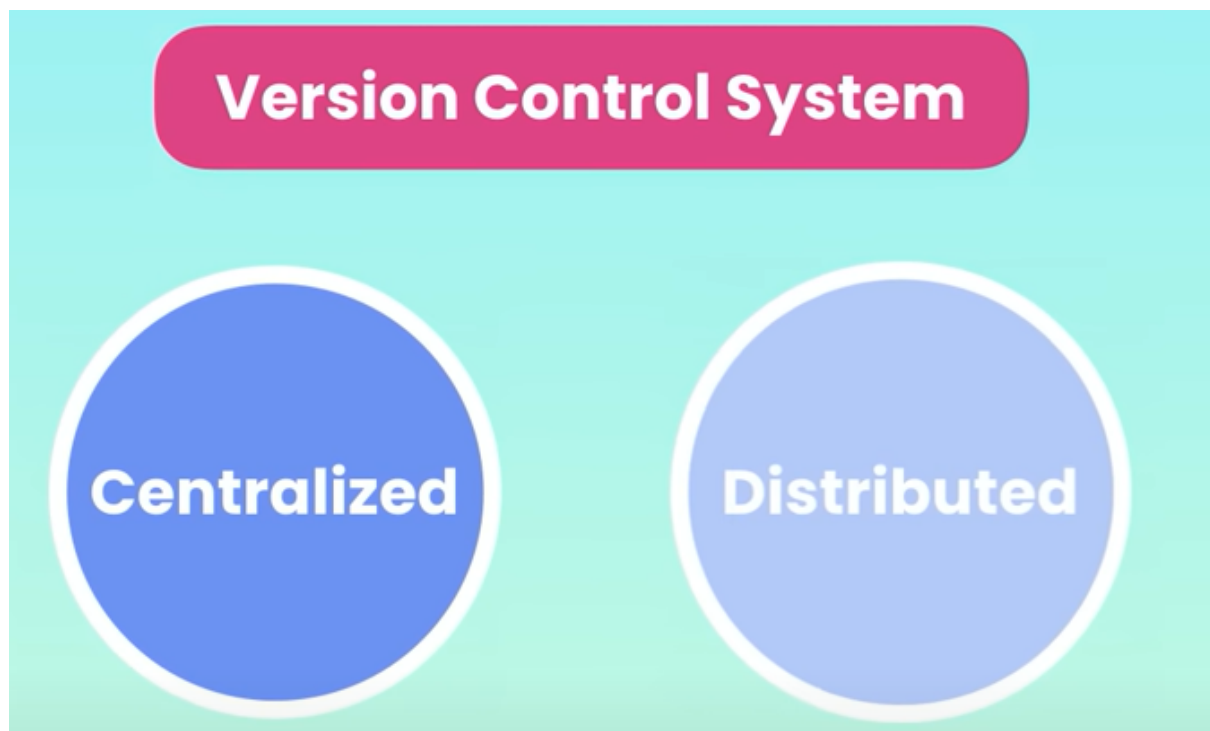
Git is the most popular and widely used version control system in the world and it is free and open source. Version control system is used to keep track of the changes that we have made to code in a special database called “repository”.

We can see the project history, who has made particular changes and why did he make changes like that. If we have made a mistake in check in then we can easily revert back our changes into earlier state.

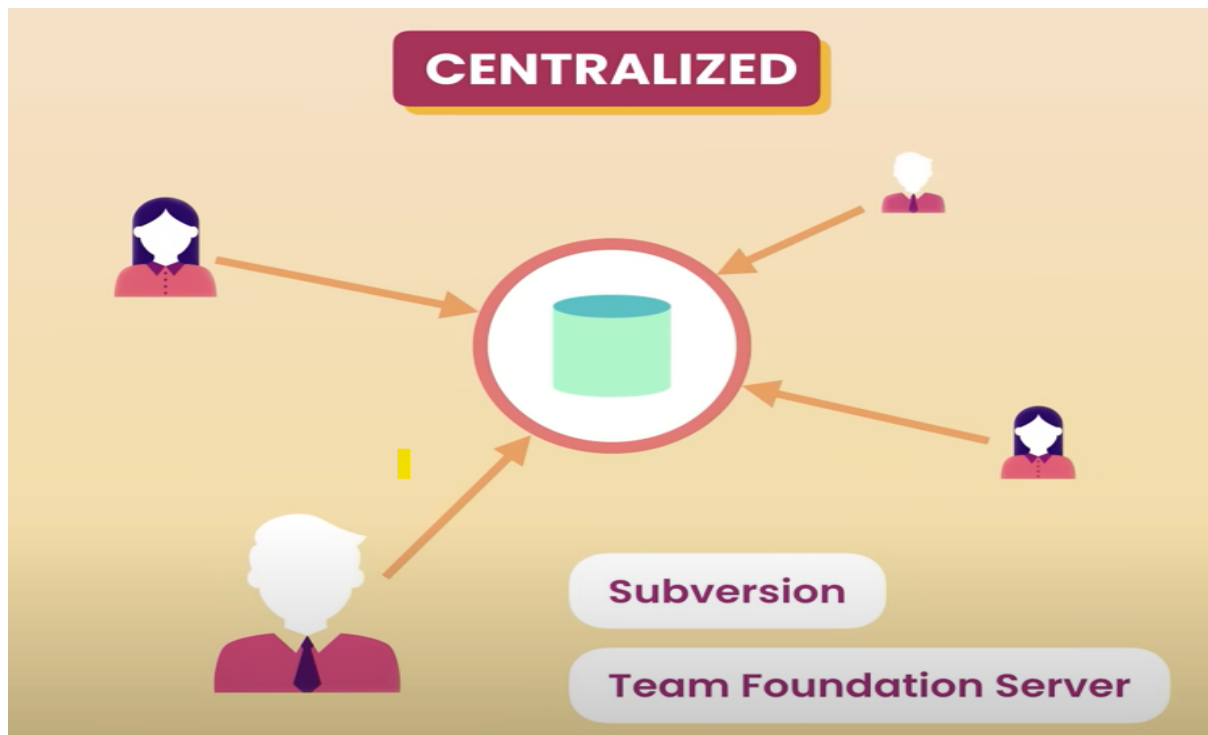


The version control system falls into two categories:

1. Centralised
2. Distributed

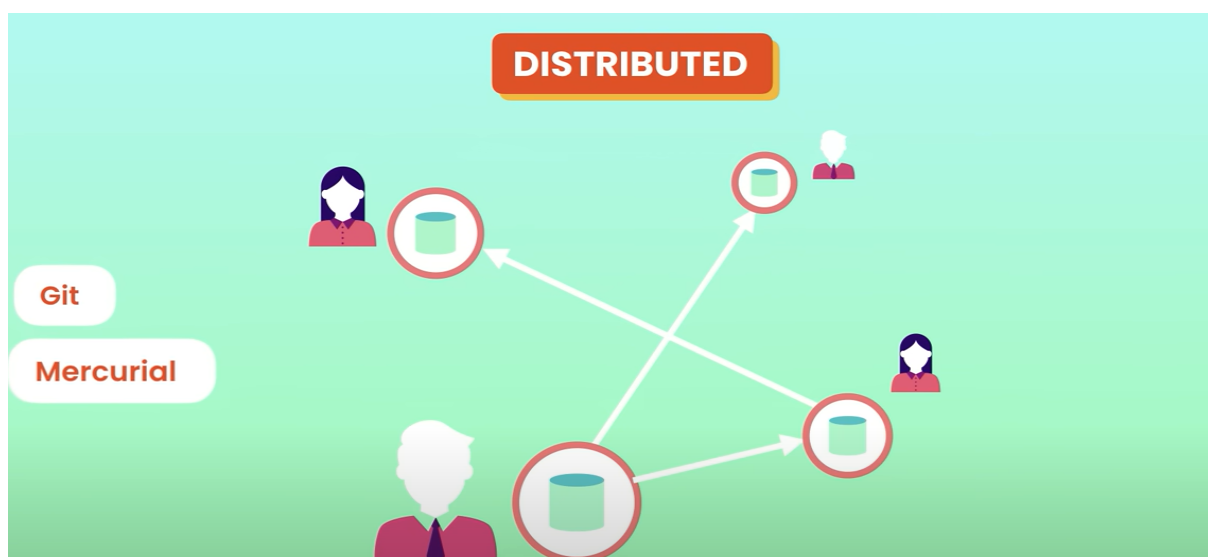


In a **centralized** system all the team members will be connected to a centralised server to get the latest copy of the code and to let everyone know about the changes in the code.



The problem with the centralized architecture is that if the server goes offline we can not merge our code or take a snapshot or get the latest project till the server comes back online.

In the distributed architecture every team member will be having a copy of the project with the history on their machine, so we do not have this problem. If the server is offline we can synchronize the changes directly with the teammates.



Git and Mercurial are the examples of the distributed version control system however the Git is the most popular version control system in the world because of the following reasons:

1. It is free to use.
2. It is open source.
3. It is super fast.
4. It is Scalable.

How to know whether the Git is installed on our machine ?

Open a command prompt and type in **git --version**. If it is not installed then we will get a error message like below

```
C:\>git --version
'git' is not recognized as an internal or external command,
operable program or batch file.

C:\>
```

Some of the terms that we should be familiar with while using the Git.

- Directory -> **Folder**
- Terminal or Command Line -> **Interface for Text Commands**
- CLI -> **Command Line Interface**
- cd -> **Change Directory**
- Code Editor -> **Word Processor for Writing Code**
- Repository -> **Project, or the folder/place where your project is kept**
- Github -> **A website to host your repositories online**
 - **Github** - Frequent confusion arises between Git and GitHub. To clarify, Git is a version control tool utilised to monitor code changes over time, effectively managing the development process. On the other hand, GitHub functions as a web-based platform where we can host and manage our Git repositories, providing a collaborative and centralized environment for teams to work together on projects.

Frequently utilized terminologies and Git commands include:

- Clone -> **Bring a repository that is hosted somewhere like Github into a folder on your local machine**
- add -> **Track your files and changes in Git**
- commit -> **Save your files in Git**
- push -> **Upload Git commits to a remote repo, like Github**
- pull -> **Download changes from remote repo to your local machine, the opposite of push**

1. **Clone** - This command is employed when a repository exists on GitHub but not on the local machine. It allows users to retrieve the repository and its contents to their local environment, enabling them to make changes and updates.

2. **Add** - When modifications are made to files, or new files are created or deleted, the "add" command informs Git to track these changes. It prepares the changes to be included in the next commit.

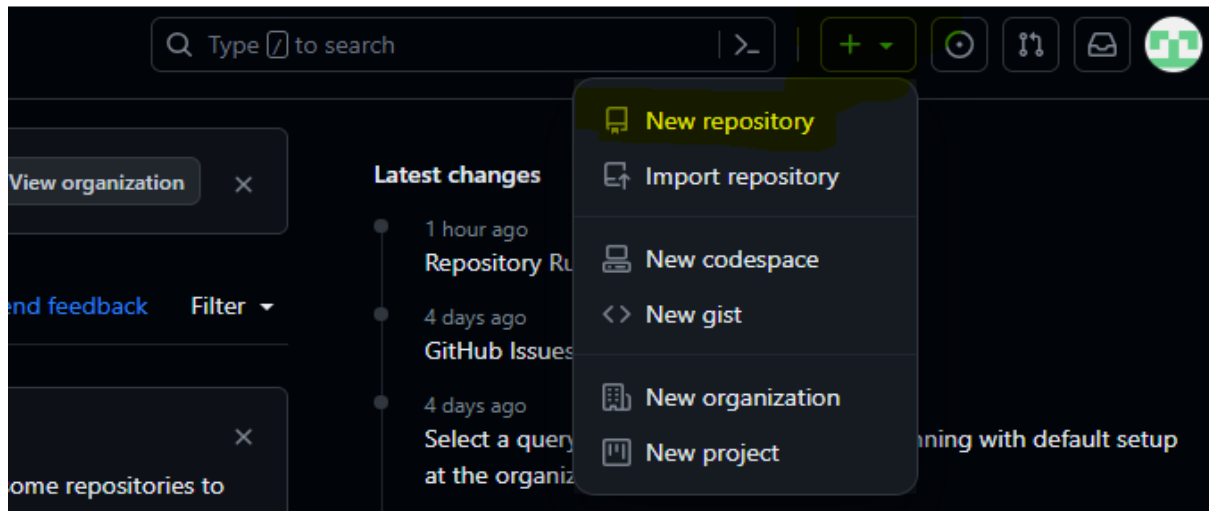
3. **Commit** - As mentioned earlier, Git's purpose is to record changes made to the codebase. The "commit" command is used to create a snapshot of the changes made since the last commit. This action effectively saves the changes to the local repository.

4. **Push** - Once changes have been committed locally and are ready to be uploaded to a remote location, such as GitHub, the "push" command is employed. It transmits the committed changes to the remote repository, making them available for others to access.

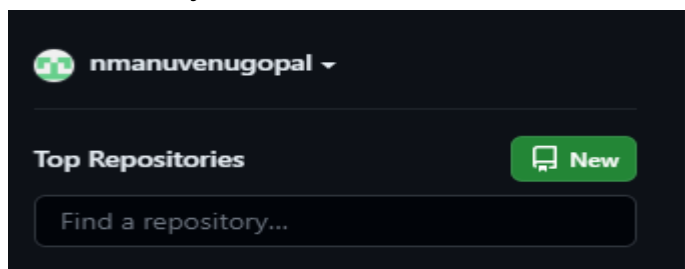
5. **Pull** - When changes have been made to the remote repository, for example, by a teammate, and these changes need to be brought into the local repository, the "pull" command is used. This action fetches and merges the changes from the remote repository into the local workspace.

The next thing we need to do is to create a repository. We can create a directory in Github using following ways:

First way is to use the “+” symbol located near to the user profile picture.



Second way is to use the “New” button on the top left corner.



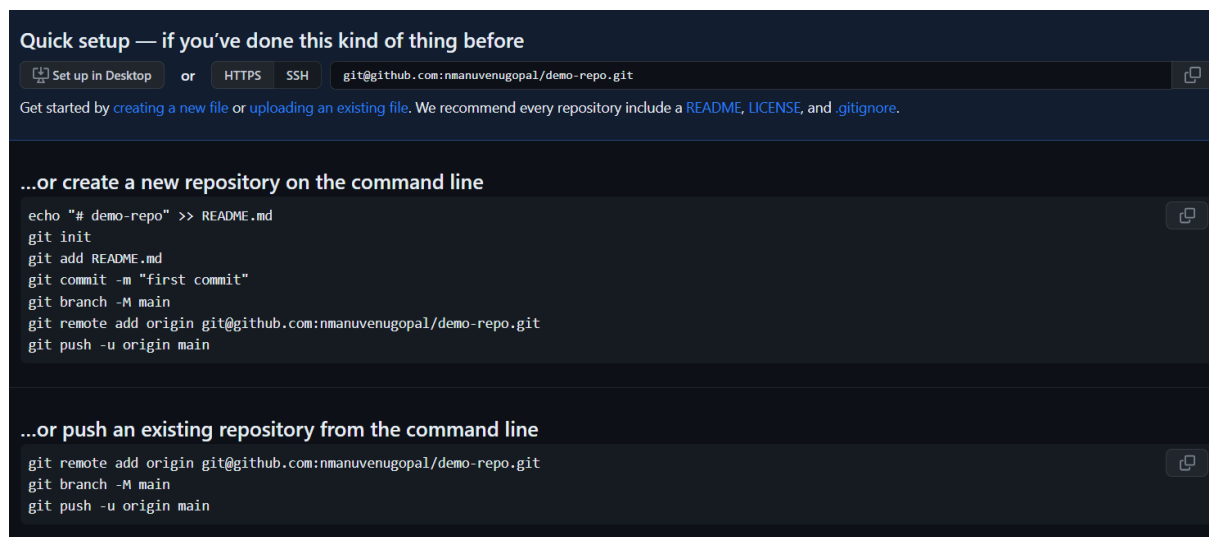
Click on the add button, fill in the details and click “Create repository” button.

A screenshot of the 'Create new repository' form in GitHub. The 'Owner' is set to 'nmanuvenugopal'. The 'Repository name' is 'demo-repo', with a green checkmark indicating it is available. Below this, a message says 'Great repository names are short and memorable. Need inspiration? How about silver-system ?'. The 'Description' field contains 'Demo for the Glthub intro'. At the bottom, there are two radio buttons: 'Public' (selected) and 'Private'. The 'Public' option has a description: 'Anyone on the internet can see this repository. You choose who can commit.' The 'Private' option has a description: 'You choose who can see and commit to this repository.'

Repository is our project, it contains all the files and subfolders that project utilises to work properly.

We can create the files and folders for the repository locally on our machine or we can create it through the online editors on the Github website.

After we click on the create repository button, I am going to create a basic readme markdown file. Readme.md is the most basic file that contains the basic information about the project, what it does or any other specific information or details.



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

Set up in Desktop or HTTPS SSH git@github.com:nmanuvenugopal/demo-repo.git

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

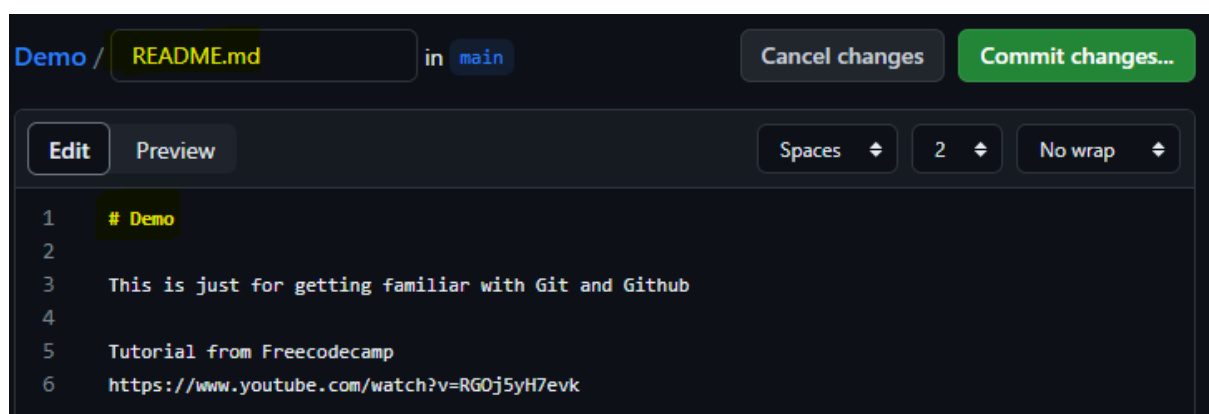
...or create a new repository on the command line

```
echo "# demo-repo" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com:nmanuvenugopal/demo-repo.git
git push -u origin main
```

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

```
git remote add origin git@github.com:nmanuvenugopal/demo-repo.git
git branch -M main
git push -u origin main
```

I have clicked on creating a new file and I need to name it as README.md file. Markdown file have many shortcuts like “#” for the main header.



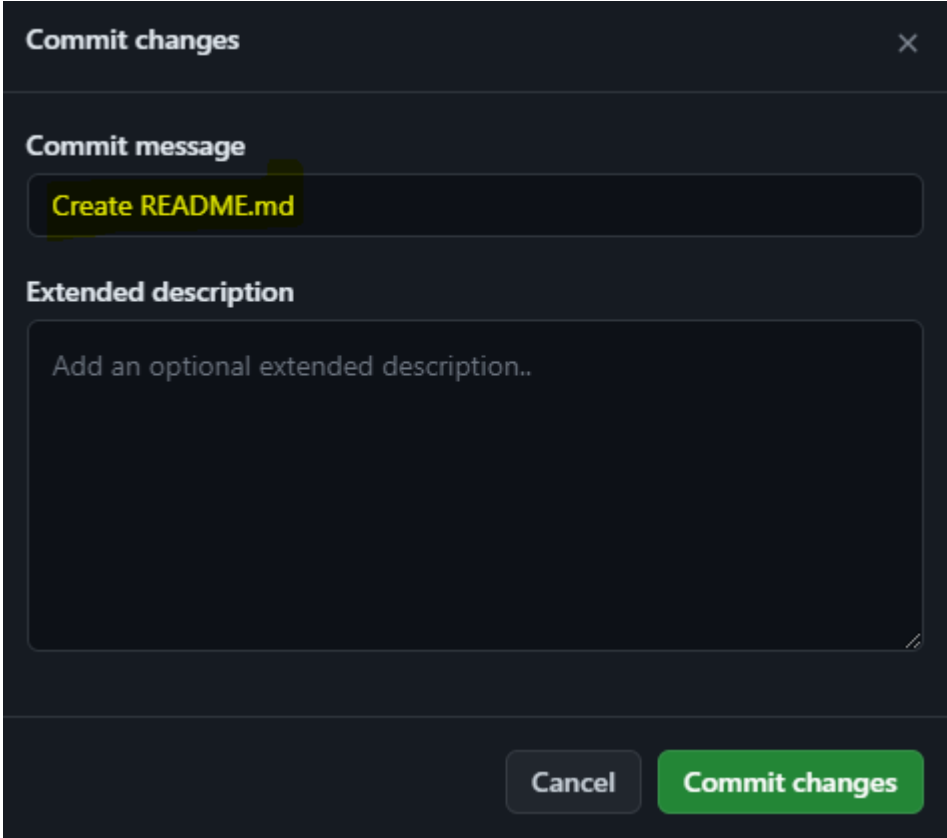
Demo / README.md in main Cancel changes Commit changes...

Edit Preview Spaces 2 No wrap

```
1 # Demo
2
3 This is just for getting familiar with Git and Github
4
5 Tutorial from Freecodecamp
6 https://www.youtube.com/watch?v=RG0j5yH7evk
```

Once I enter the details about the project in the README.md file, the next thing will be committing the changes we have made. For that we need to click on the commit change button.

Upon clicking the commit changes button, a dialog box will appear:

A dark-themed dialog box titled "Commit changes" with a close button (X) in the top right corner. It contains two main sections: "Commit message" and "Extended description". The "Commit message" section has a text input field with the text "Create README.md" entered. The "Extended description" section has a larger text area with the placeholder text "Add an optional extended description..". At the bottom of the dialog, there are two buttons: "Cancel" and "Commit changes".

Commit changes [X]

Commit message

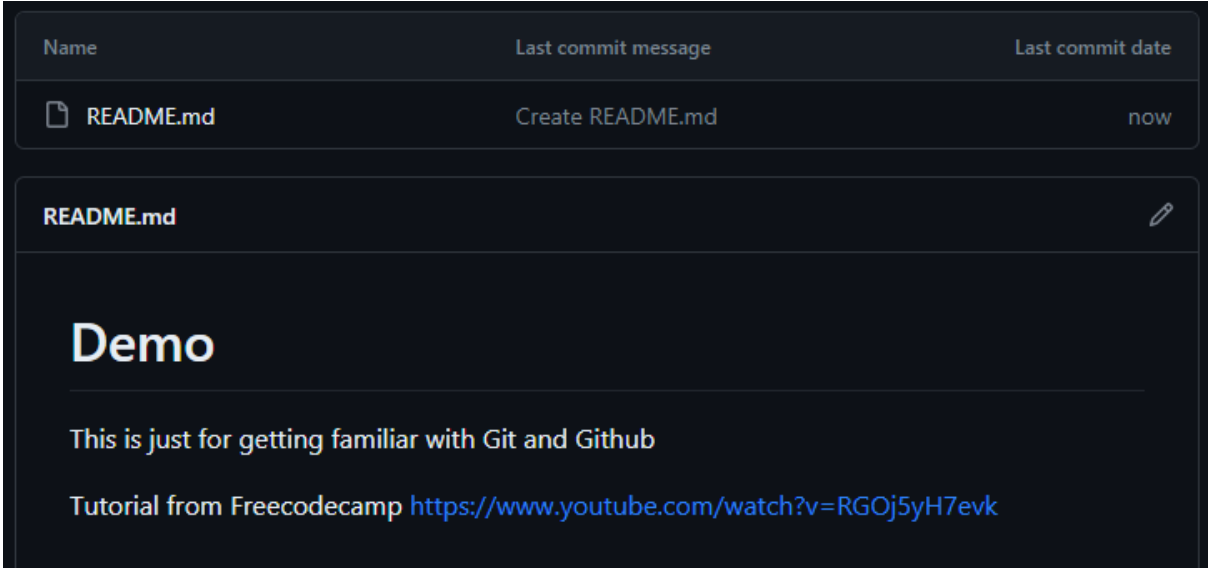
Create README.md

Extended description

Add an optional extended description..

Cancel Commit changes

We can add or update the details and click on the commit changes button for saving our changes.

A dark-themed interface showing a file list and the content of a selected file. The file list at the top has columns for "Name", "Last commit message", and "Last commit date". Below it, a file named "README.md" is listed with the commit message "Create README.md" and the date "now". Below the file list, the content of "README.md" is displayed, including a title "Demo", a subtitle, and a link to a tutorial from Freecodecamp.

Name	Last commit message	Last commit date
📄 README.md	Create README.md	now

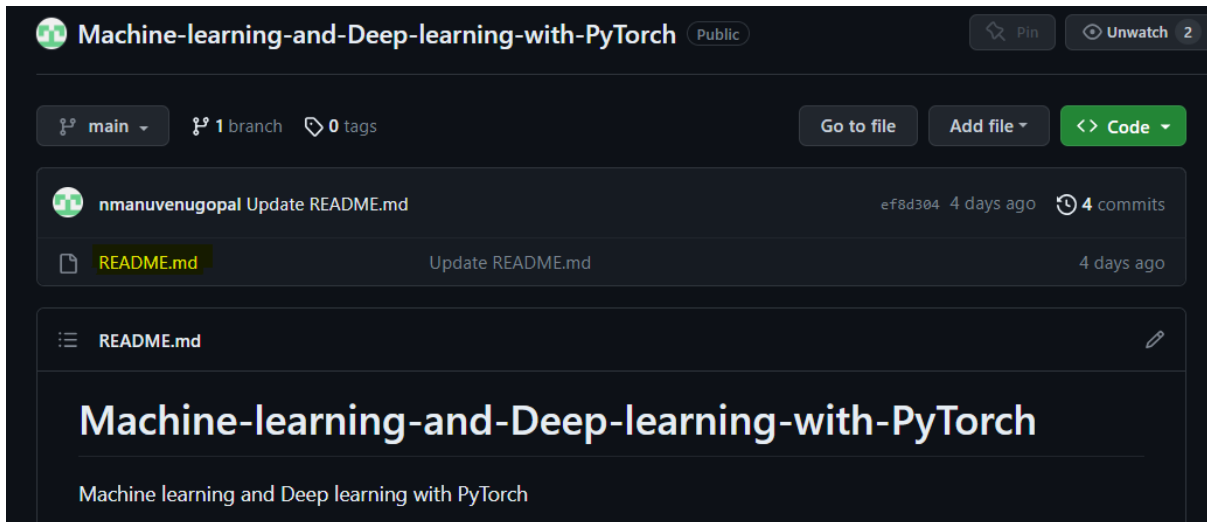
README.md [edit icon]

Demo

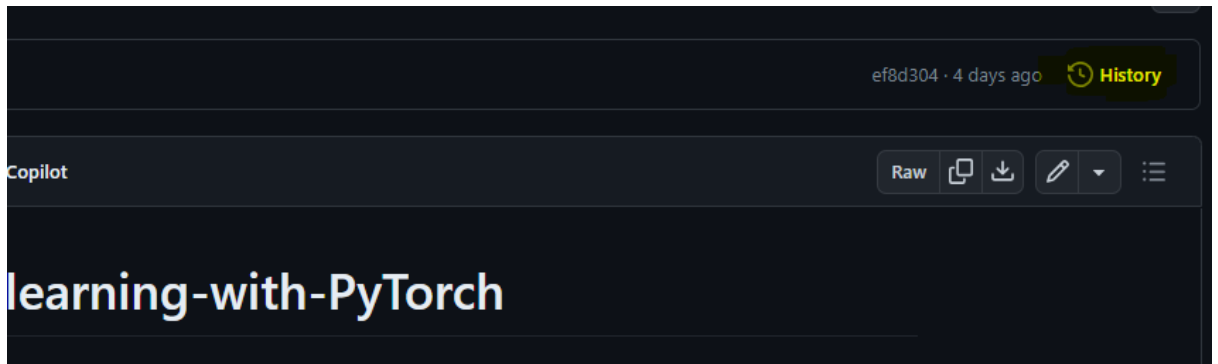
This is just for getting familiar with Git and Github

Tutorial from Freecodecamp <https://www.youtube.com/watch?v=RGOj5yH7evk>

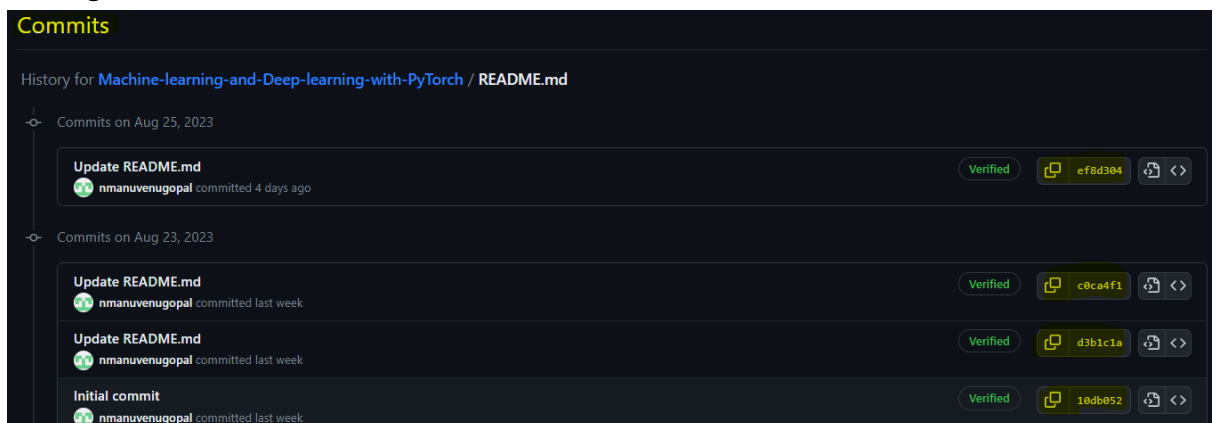
Suppose if we want to see the changes made to the README file then



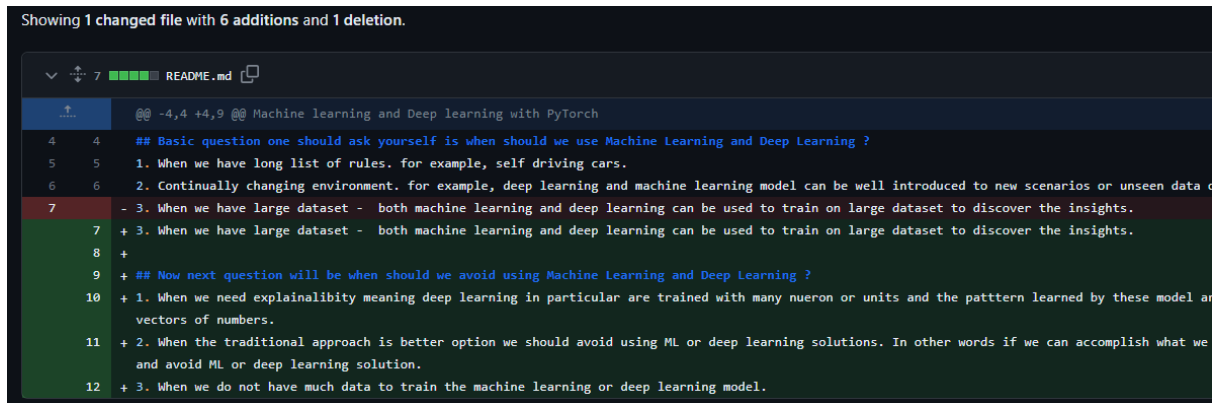
1. Click on the file for which we need to see the history of changes.



2. On the top right corner, we can see the history option, then we can see the list of changes and unique id associated with every change.



3. If we click on a item above, then we can see the changes corresponding to that commit



We need to create SSH and KEY for performing the various operations in Git. For setting up an SSH key kindly follow the steps in below link:
<https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent?platform=windows>

I have done this through the Git Bash(if using windows it is recommended)

```

PS-manu.venugopalan@LAP1097-MINGW64 /c/Git_Practise
$ ssh-keygen -t ed25519 -C "nmanuvenugopal@outlook.com"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/c/Users/manu.venugopalan/.ssh/id_ed25519): /c/Users/manu.venugopalan/.ssh/id_window_git
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/manu.venugopalan/.ssh/id_window_git
Your public key has been saved in /c/Users/manu.venugopalan/.ssh/id_window_git.pub
The key fingerprint is:
SHA256:pg9GHtsJ1Rkr553WT+G1k8VMrjXX2ZtUw0BpEHkyhiM nmanuvenugopal@outlook.com
The key's randomart image is:
---[ED25519 256]---
      oo=0+.o|
    E + 0 + **|
      + B =  =@|
      . + . oo*B|
      + S . + **+|
    o B . . + |
      * o      |
      . o      |
      .        |
-----[SHA256]-----

```

```

RS+manu.venugopalan@LAP1092 MINGW64 /c/Git_Practise
$ ssh-add ~/.ssh/id_window_git
Identity added: /c/Users/manu.venugopalan/.ssh/id_window_git (nmanuvenugopal@outlook.com)

RS+manu.venugopalan@LAP1092 MINGW64 /c/Git_Practise
$ cat /c/Users/manu.venugopalan/.ssh/id_window_git.pub
ssh-ed25519 AAAAC3NzaC1TdZDI1NTE5AAAAIJXqfABkfquhAE61QmXM80S4v2UnAkmCi9bpTVLzsMx+H nmanuvenugopal@outlook.com

```

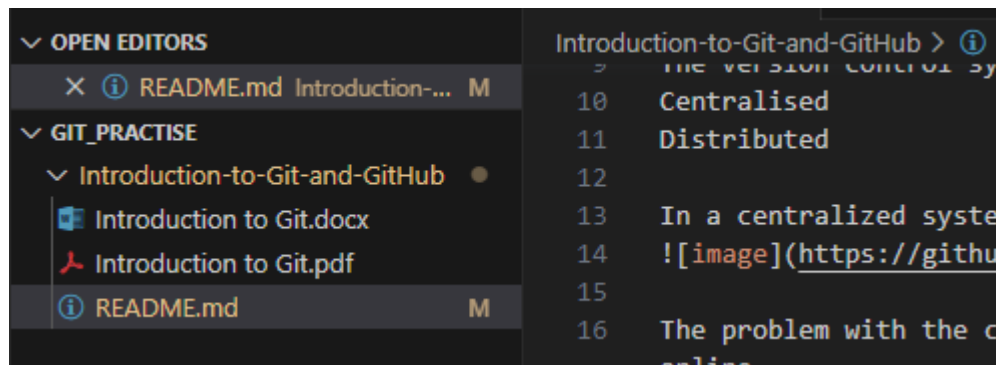
For getting familiar with Git we need to have experience or hands-on experience.

1. I have created an empty folder in my local drive and opened it in VS Code.
2. After opening the empty folder “Git_Practise”, now a new terminal window in VS code.

3. Now we can use the git clone command to import the code from the github to the local.

```
RS+manu.venugopalan@LAP1092 MINGW64 /c/Git_Practise
$ git clone git@github.com:nmanuvenugopal/Introduction-to-Git-and-GitHub.git
Cloning into 'Introduction-to-Git-and-GitHub'...
remote: Enumerating objects: 13, done.
remote: Counting objects: 100% (13/13), done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 13 (delta 2), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (13/13), 2.15 MiB | 3.03 MiB/s, done.
Resolving deltas: 100% (2/2), done.
```

4. After these steps, if we go back to the VS Code then we can see that the files in the corresponding repo are downloaded to our practice folder.



5. I have made the changes to the readme file via VS Code and need to know the details of the files I changed, for this we can use the “status” command.

```
RS+manu.venugopalan@LAP1092 MINGW64 /c/Git_Practise/Introduction-to-Git-and-GitHub (main)
$ 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")
```

6. What happens if I add a new file, let it be an example.html file to the repository via VS Code then the “git status” returns the following. Status says that example.html is not tracked, that means it is a new file (Git doesn't know about the file yet) and I have tried to modify the readme file too.



```
RS+manu.venugopalan@LAP1092 MINGW64 /c/Git_Practise/Introduction-to-Git-and-GitHub (main)
$ 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)
    example.html

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

7. For adding the files we can use the “add” command and there are two ways in which we can use the add method. Add with the specific file name or Add with period (‘.’).

```
RS+manu.venugopalan@LAP1092 MINGW64 /c/Git_Practise/Introduction-to-Git-and-GitHub (main)
$ git add example.html

RS+manu.venugopalan@LAP1092 MINGW64 /c/Git_Practise/Introduction-to-Git-and-GitHub (main)
$ 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:   example.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
```

```
RS+manu.venugopalan@LAP1092 MINGW64 /c/Git_Practise/Introduction-to-Git-and-GitHub (main)
$ git add .

RS+manu.venugopalan@LAP1092 MINGW64 /c/Git_Practise/Introduction-to-Git-and-GitHub (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

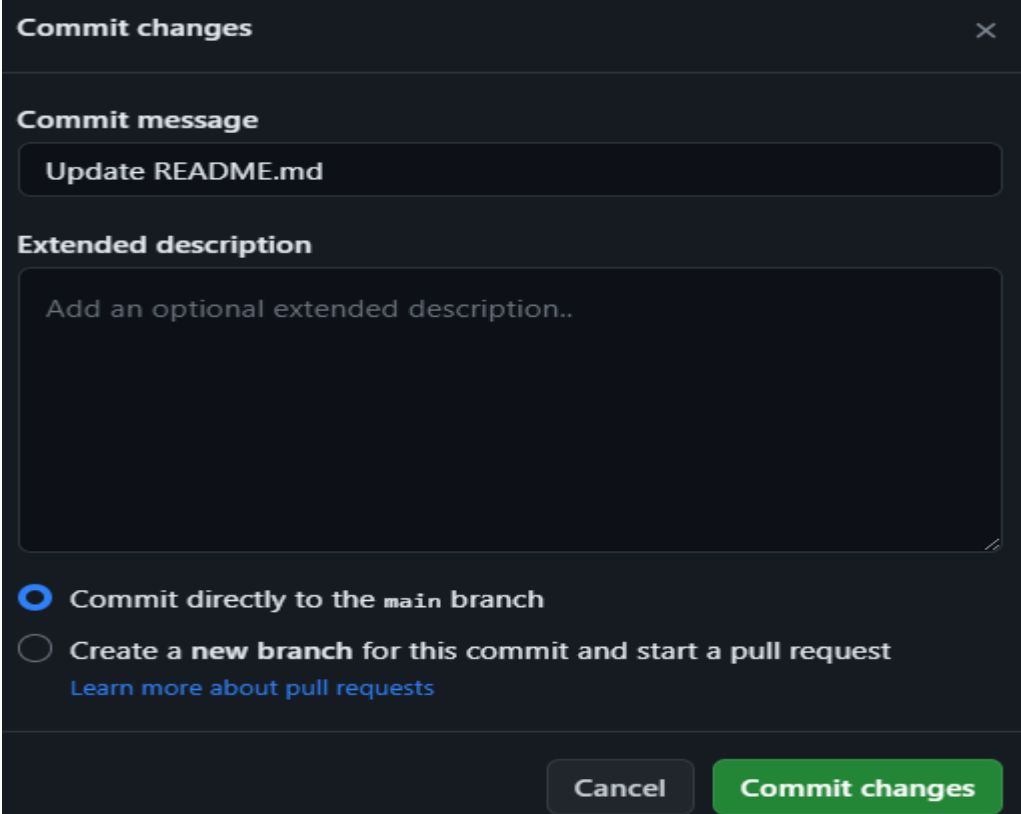
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   README.md
    new file:   example.html
```

If we are using period (.) then we are saying the git to track all of the newly added and updated records or files. In our case if we add a period then both readme and example.html will be staged with git. But in order to experiment i have used both add with specific file name and add with period too.

8. Now we can see that all of the updated and untracked files are now staged to git and they are ready to be committed. For committing the code we use the commit command.

```
PS+manu.venugopalan@AP1092-MINGW64 /c/Git_Practise/Introduction-to-Git-and-GitHub (main)
$ git commit -m "added example.html" -m "added example.html file for demonstration and updated readme file to include reference"
[main 3a728ff] added example.html
2 files changed, 5 insertions(+)
create mode 100644 example.html
```

First description is for the commit message in the commit command and the second description is for extended description.



Commit changes

Commit message

Update README.md

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

9. We have still saved our code locally, but the commit isn't live in GitHub yet. For making it live in GitHub we use another command "push". Push command will push the changes to a remote repository where my project is hosted.

Reference

Git and Github for beginners - freeCodeCamp

<https://www.youtube.com/watch?v=RGQj5yH7evk>

Git Tutorial for Beginners: Learn Git in 1 Hour

<https://www.youtube.com/watch?v=8JJ101D3knE&t=880s>