Version Control

It also called as Source Control.
It is a System which record changes to your files so later on you can manage them as specific Version.
It also allows multiple developers to collaborate on one project.
Why?

- track the code
- manage the changes of code.
- team can collaborate with each other
- revert the previous version if is there any bug in latest version.
- we ca do branching and merging for development
- check the entire history like who changed the code, what feature added and why?

Types?

Local version Control System:

Stores version of files locally in local-disk
Simple but not good for collaboration.
example: RCS (revision VCS)

Centralized Version Control System:

- one central server stores all versions
- developers can check code, make changes can go back.
- Easy to understand, good to work with teams for same network.
- problem is: single point failure.
- example: SVN (Sub versions), CVS

Distributed Version Control System:

- every developer can have full copy of repository
- supports offline work
- faster development, good for collaboration
- example: GIT

To work with the Git, it must be downloaded to your system.
https://git-scm.com/downloads (download Link)
follow the simple installation process and install.
you can run in Git Bash or CMD
open cmd and run git version command

```
C:\Users\NEW>git version
git version 2.41.0.windows.3
```

or else you can right click in your system and click on open git bash here.

```
NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop
$ git version
git version 2.41.0.windows.3
```

GIT Terminology

Repository (Repo): its a folder which contains all your project files and folders also have all the history of changes.

Commit: Snapshot of changes made to your codebase with some message.

Branch: separate line of developer, used for development, bug fix etc.

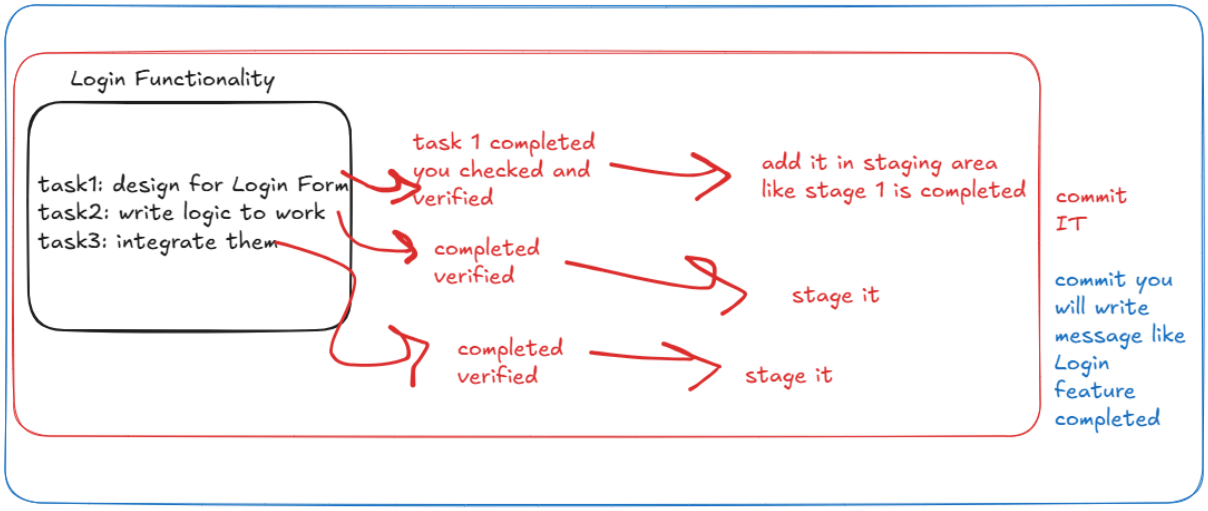
Merge: combining one branch to another branch for integrating features.

clone: Copy entire remote code to local machine

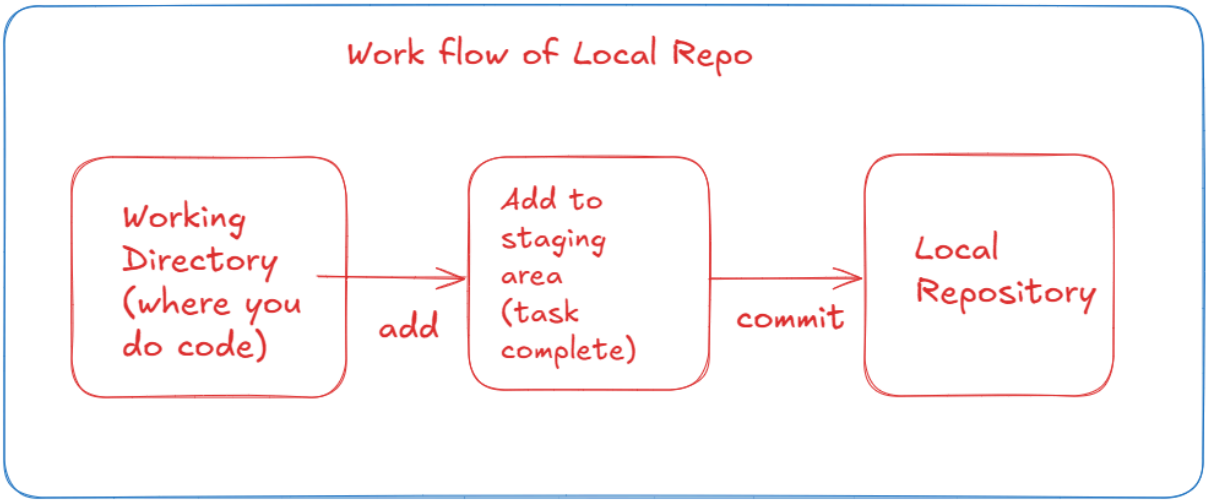
pull: fetch or merge changes from remote to local, merge changes in remote repo.

push: upload local commit to remote

Staging area: a middle ground where you set up your changes before commit.



Work Flow



When we start working for project its just a folder means its called Working Directory.
When you initialize git Repo in that folder it becomes Git Repository which is having .git hidden folder inside the same.
This folder keep the track of staging area commits etc...
and that's called Local Repository.
Let's create folder, one file inside the same. Open Git bash,

```
NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop
$ mkdir myproject

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop
$ cd myproject

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/myproject
$ nano file1.txt

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/myproject
$ cat file1.txt
Hello All this is my Sample Git project

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/myproject
$ ls
file1.txt
```

This is working directory, to make it git repo execute git init command.
when this command executed it will initialize empty git repo and create .git hidden folder to your repo.

```
NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/myproject
$ git init
Initialized empty Git repository in C:/Users/NEW/Desktop/myproject/.git/
```

```
NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/myproject (master)
$ ls -a
./  ../  .git/  file1.txt
```

here you can see, after initialization, its also showing master branch by default.

.git folder is keeping track of all staging area and commits.

```
NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/myproject (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file1.txt

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

No to Track the untrack File use add command.

```
NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/myproject (master)
$ git add file1.txt

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/myproject (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   file1.txt
```

Now the file is tracked.

Remove it from staging area

```
NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/myproject (master)
$ git rm --cached file1.txt
rm 'file1.txt'

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/myproject (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file1.txt

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

--cached means remove from staging but keep it in working directory.

If you want to stage all files then use git add . (. indicates all files and folders)

```
NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/myproject (master)
$ git add .

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/myproject (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   file1.txt
```

Lets' Commit

```
NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/myproject (master)
$ git commit -m "Feature 1 completed"
[master (root-commit) 26ef93c] Feature 1 completed
 1 file changed, 1 insertion(+)
 create mode 100644 file1.txt

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/myproject (master)
$ git log
commit 26ef93c16e1b9470347ec66911352ad2ed01c8e5 (HEAD -> master)
Author: sonam-niit <sonam.gravity@gmail.com>
Date:   Sat Apr 12 10:18:17 2025 +0530

    Feature 1 completed
```

While doing commit if you are getting config error then we need to config git user to local system.

we need to configure git User

3 Types:

System Configuration:

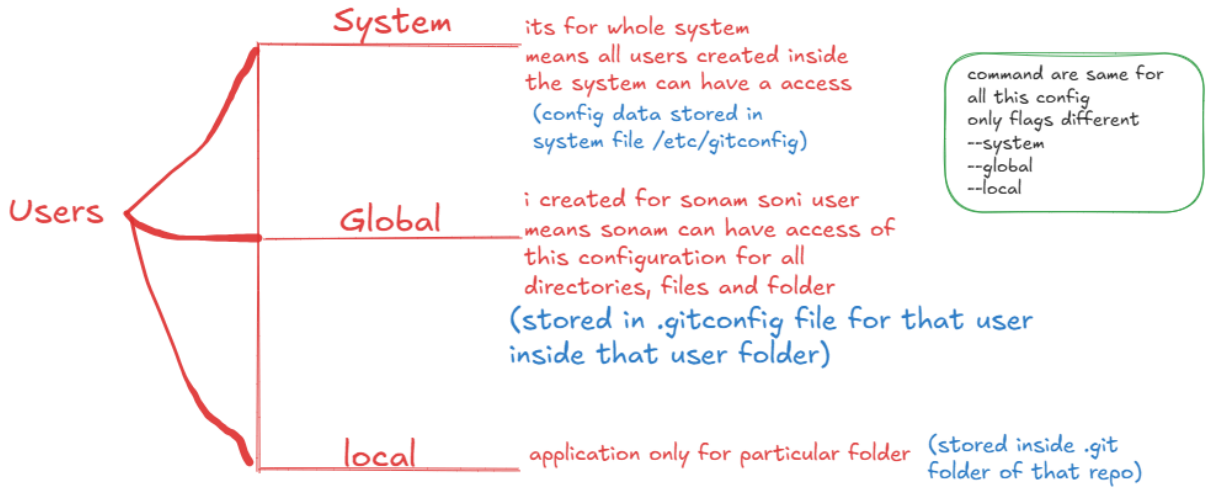
Applied to all users of a system

Global Configuration:

Applicable to to single user but for all files and folders.

Local Configuration:

For only 1 folder




```
NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/myproject (master)
$ git config --global user.name "sonam-niit"

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/myproject (master)
$ git config --global user.email "sonam.gravity@gmail.com"

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/myproject (master)
$ git config --global --list
user.email=sonam.gravity@gmail.com
user.name=sonam-niit
```

For few commands you want to set alias.

```
NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/myproject (master)
$ git config --global alias.st status

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/myproject (master)
$ git config --global alias.co commit

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/myproject (master)
$ git config --global alias.ci commit

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/myproject (master)
$ git config --global alias.co checkout

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/myproject (master)
$ git config --global alias.br branch

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/myproject (master)
$ git config --global --list
user.email=sonam.gravity@gmail.com
user.name=sonam-niit
alias.st=status
alias.ci=commit
alias.co=checkout
alias.br=branch
```

Changing Default branch as main branch

```
NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/myproject (master)
$ git branch -M main

NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/myproject (main)
$ git log
commit 26ef93c16e1b9470347ec66911352ad2ed01c8e5 (HEAD -> main)
Author: sonam-niit <sonam.gravity@gmail.com>
Date:   Sat Apr 12 10:18:17 2025 +0530

    Feature 1 completed
```

Now we want to push this on remote Repository.

Create Repository on GitHub

To push Local Repo to Remote we need to set Remote Origin:

git remote add origin origin-link (command to set origin)

origin you need to set only once

```
NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/myproject (main)
$ git remote add origin https://github.com/sonam-niit/PWProject.git
```

Now to push the code: git push -u origin main

means I want to push my commits to mentioned origin on main branch

origin means where to push code: its your remote repository link
-u is to set upsteam banch means if i say -u origin main means it will set upsteam branch to main
next time if i say git push that means it will push the code of main branch to remote

```
NEW@DESKTOP-4F8ELLU MINGW64 ~/Desktop/myproject (main)
$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 260 bytes | 65.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/sonam-niit/PWProject.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

If you are doing this for very first time then, it will ask for browser authentication.
say Yes for it and you will be redirected to your browser then enter credentials, once login done you can push the code.

If you are maintaining repository from local then we need to initialize it local, manage locally, connect to remote by setting up remote origin and then we can do push command execution.

create folder --> mkdir project
- move --> cd project
- create file --> nano file.txt
- initialize repo --> git init
- stage file --> git add file.txt
- check stattus --> git status
- commit the staged content --> git commit -m "first commit"
- check the commit history --> git log
- set master branch to main as default --> git branch -M main
- to push them remotely --> create remote repository by going to github --> new repo --> add name + description --> not check on readme creation --> you will get remote origin set up command
- set origin --> git remote add origin origin link
- push the code: --> git push -u origin main

Also- for the very first time if you trying to commit or push it can give error for configuration so set up configuration only once.
- git config --global user.name "user-name"
- git config --global user.email "user@email.com"

this is one time setup after this i want to add some more codes and i want to push then how?
just do : git add . (. indicates all updated files in working directory)

git commit -m "new version update"
git push
just 3 commands.