



Lets GIT it



--Pranoy Patra

Why GIT?



We are Unorganised in nature. Isn't it?

Now thinks about Developers 🙄

They have to handle a large number of files
and they have to update it continuously 🙄

It is very difficult for them to remember the changes 🙄





Hey !
I Am Here To
Help You



Git : Git is a distributed version control system

Now we can use Git to solve our problem to organize files properly

Lets see the Benefits

Benefits of Git

- Easily recover files
- Who introduced issue and when
- Rollback to previously working state

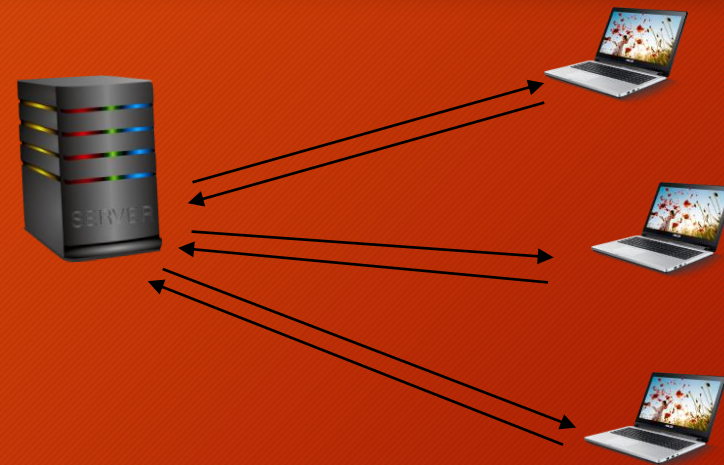
History of VCS(Version Control System)

Local VCS

- Uses DB to keep track of updates
- Can track files & Rollback
- If you loose your hard disk you loose everything

Centralized VCS

- Uses Remote Server to keep data
- Can track files & Rollback
- Better than Local VCS
- Anyone who have access can contribute to the project
- If Server affected you may loose the previous versions of your code



Distributed VCS

- Same as Centralize VCS +
- In every users computer previous versions of the projects will store

What is GitHub



GitHub is a website which hosts Git Repositories

Features of Git

- Captures Snapshots not Differences
 - Light weight
- Almost Every operations is local
 - You can work without internet until you need to push the files
- Git has Integrity
 - When you transfer files from your computer to Remote server or vice versa then Git takes care of your files using **Checksum** method.
 - Your files should be safe and not to be changed by Hacker
- Git generally only ads data
 - As frequent you change datas in repository your .git folder's size will increase

Checksum Method:

Every Files has its Unique Fingerprint (SHA1 value).So if the file changes the value of fingerprint will change it doesn't matters how small the change is.

So if two fingerprints are same (When Sending & When Receiving) then we we will sure that we have received our original files

Install Git on Local System

[Download Git](#)

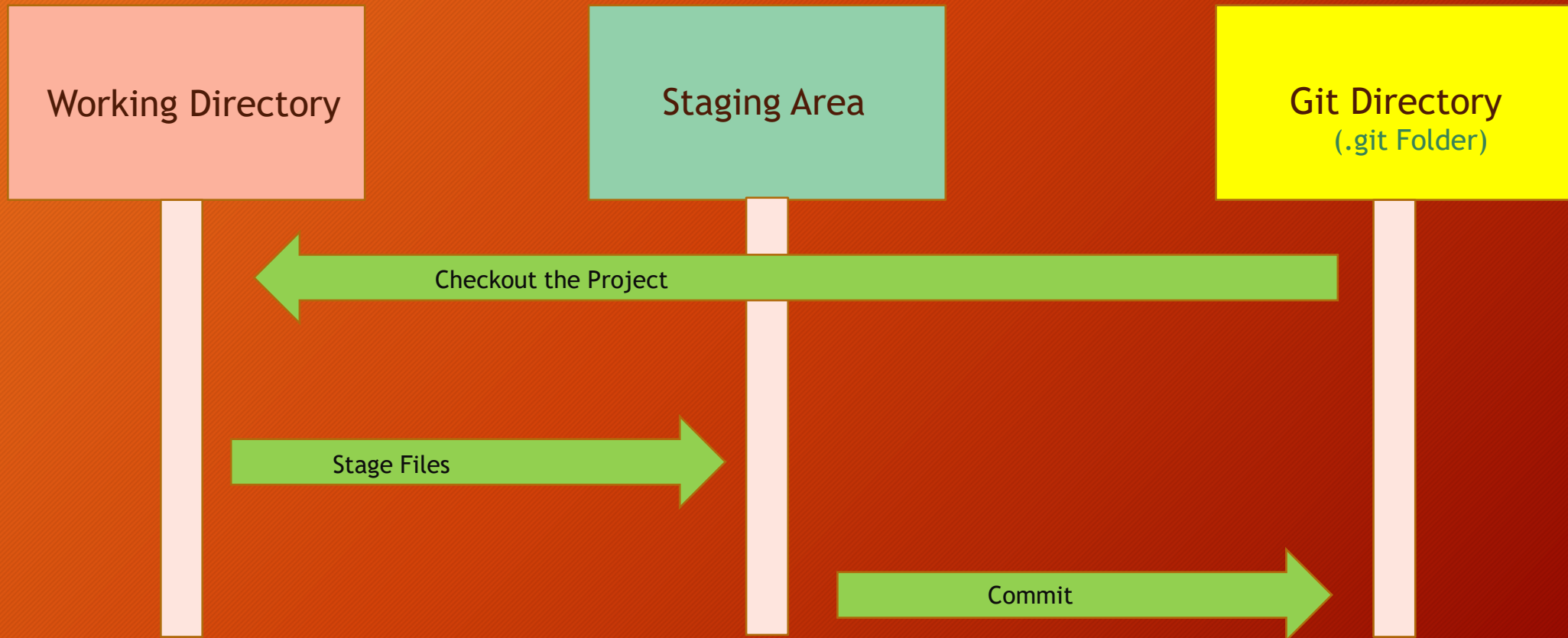
Choose your system comfortable version
And download it

You will get Git Bash and Git GUI with it

Some Useful Linux Commands

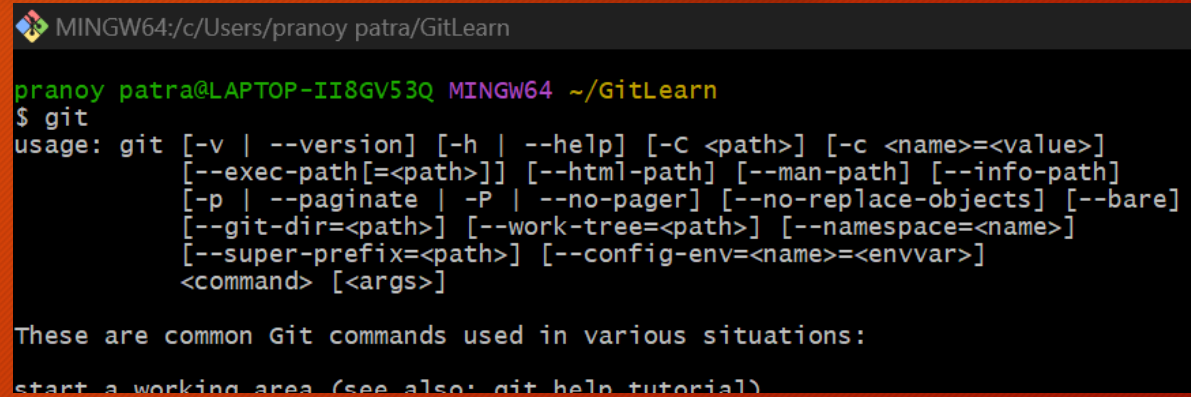
- To open folder `cd "folder_path or name"`
- Back `cd`
- See Folder and Files list `ls`
- Paste **Shift+insert**
- Present work directory `pwd`
- To create an empty file `touch "filename.txt"`

Git - Three Stage Architecture



Git Bash Commands

- `git` (It will gives you an overview)
- `git status` (To check the status of git)
- `config` (Configure git for first time or update user datas)
 - `git config --global user.name="Your Name"` (Set your name)
 - `git config --global user.email="Your mail id"` (Set your mail)
 - `git config user.name` (Show your name)
 - `git config user.email` (Show your Email id)
- `git init` (To initialize Git by making .git folder)
- `git add -a` (To Track All Files)
- `git add .` (To Track All Files)
- `git add "filename"` (To Track a single Files)



```

MINGW64:/c/Users/pranoy patra/GitLearn

pranoy patra@LAPTOP-II8GV53Q MINGW64 ~/GitLearn
$ git
usage: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]
          [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
          [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
          [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
          [--super-prefix=<path>] [--config-env=<name>=<envvar>]
          <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
  
```


- `git commit -m "Your Message"` (To Commit Staged Files)
(Use `-m "message"` otherwise it will open vim editor)
- `git log` (To see the logs of your Commits with details)
- `rm -rf .git` (To remove .git folder)(Be carefull about it ,
it will delete all your previous versions)

Clone a Repository

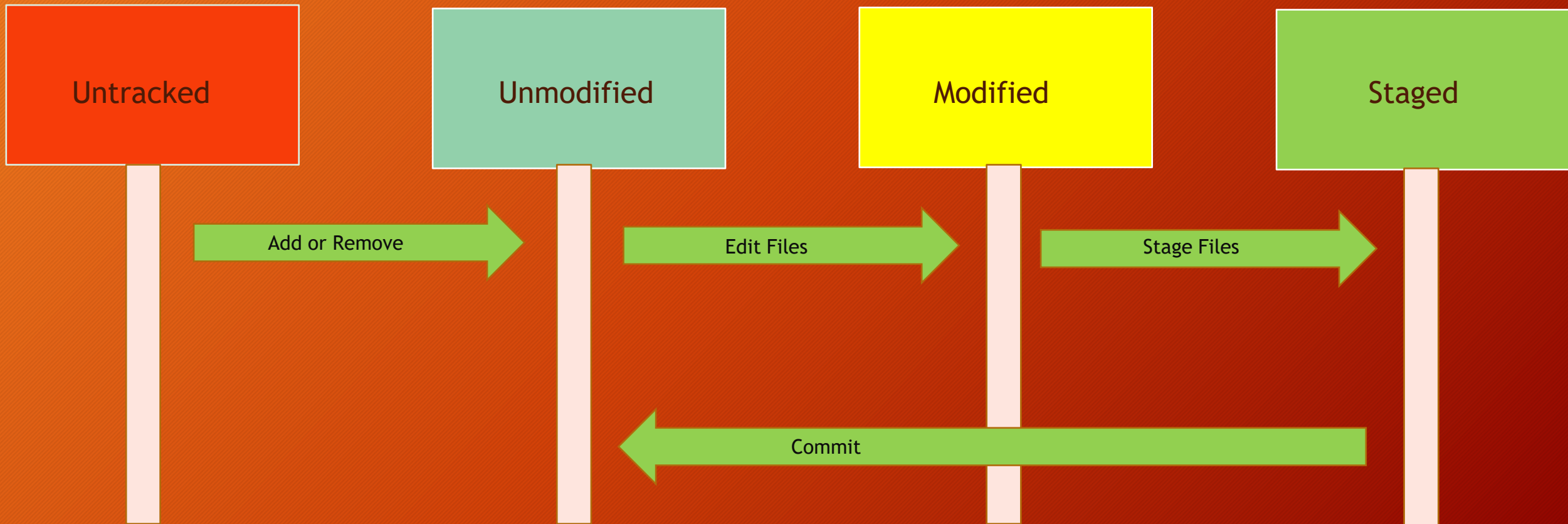
You will Clone the Project from Remote Server to Your Local Computer

Open your **GitBash** and type the Command

git clone “url of repository”
Or git clone “url of repository” “coustom name”

```
pranoy patra@LAPTOP-II8GV53Q MINGW64 ~/GitLearn (master)
$ git clone https://github.com/pranoy1171999study/analog_clock
Cloning into 'analog_clock'...
remote: Enumerating objects: 28, done.
remote: Counting objects: 100% (28/28), done.
remote: Compressing objects: 100% (21/21), done.
remote: Total 28 (delta 2), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (28/28), 109.41 KiB | 117.00 KiB/s, done.
Resolving deltas: 100% (2/2), done.
```

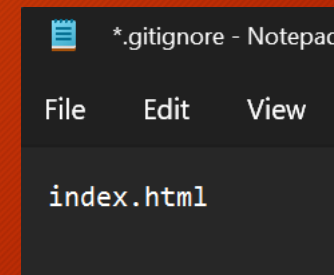
Git - File Status LifeCycle



Git .gitignore

If you don't want to track your all files (like app.log,etc)
 .gitignore is here to help you

- Generate .gitignore file
 - touch .gitignore
- Writes some files or folders inside it(Whoom to be ignored)
 - Ex: app.log,error.log..
- Add .gitignore to stage and Commit
 - git add .gitignore
 - Git commit -m "custom massage"



```

MINGW64/c/Users/pranoy_patra/GitLearn/analog_clock

pranoy_patra@LAPTOP-II8GV53Q MINGW64 ~/GitLearn/analog_clock (master)
$ touch .gitignore

pranoy_patra@LAPTOP-II8GV53Q MINGW64 ~/GitLearn/analog_clock (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   css/styles.css
    new file:   img/clock1.png
    new file:   index.html
    new file:   js/script.js

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

pranoy_patra@LAPTOP-II8GV53Q MINGW64 ~/GitLearn/analog_clock (master)
$ git add .gitignore

pranoy_patra@LAPTOP-II8GV53Q MINGW64 ~/GitLearn/analog_clock (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   .gitignore
    new file:   css/styles.css
    new file:   img/clock1.png
    new file:   index.html
    new file:   js/script.js

pranoy_patra@LAPTOP-II8GV53Q MINGW64 ~/GitLearn/analog_clock (master)
$ git commit -m "Git ignore created"
[master (root-commit) 2a8a8c2] Git ignore created
5 files changed, 104 insertions(+)
create mode 100644 .gitignore
create mode 100644 css/styles.css
create mode 100644 img/clock1.png
create mode 100644 index.html
  
```

Git diff

Git diff will compare all the differences between your Working Directory (or last Commit)& Staging Area

git diff (Compare Working Directory & Staging Area)

git diff -staged (Compare Last Commit & Staging Area)

```
pranoy patra@LAPTOP-II8GV53Q MINGW64 ~/GitLearn/analog_clock (master)
$ git diff
diff --git a/.gitignore b/.gitignore
index 64233a9..02dbc29 100644
--- a/.gitignore
+++ b/.gitignore
@@ -1,1 @@
- index.html
\ No newline at end of file
+ hi.html
\ No newline at end of file

pranoy patra@LAPTOP-II8GV53Q MINGW64 ~/GitLearn/analog_clock (master)
$ git diff --staged
diff --git a/.gitignore b/.gitignore
index e69de29..64233a9 100644
--- a/.gitignore
+++ b/.gitignore
@@ -0,0 +1 @@
+ index.html
\ No newline at end of file

pranoy patra@LAPTOP-II8GV53Q MINGW64 ~/GitLearn/analog_clock (master)
$
```

Some More Comments

- Skipping Staging area and Commit
 - `git commit -a -m "Cousom Massage"`
- Remove & Rename
 - `git rm "file.txt"` (To Delete)
 - `git mv "file.txt" "newNeme.txt"` (To Rename)
- Untrack a File
 - `git rm --cached "file.txt"`

Git Log

You can see all your project logs here (All commits..)

(Press q to exit)

- `git log` (Show all Logs)
- `git log -p` (Show all Details Line by Line)
- `git log -p -n` (Show all Details of n (1,2,3..n) commits)
- `git log --stat` (Show Stat in short)
- `git log --preety =1` (Show all commits (1 commit in 1 line))
- `git log --preety =sort` (Show Commits in Short)
- `git log --preety =full` (Show Commits in Full)
- `git log -since=n.days` (Show commits of last n days)
- `git log -since=n.weeks` (Show commits of last n weeks)
- `git log -since=n.months` (Show commits of last n months)
- `git log -since=n.years` (Show commits of last n years)

Git Log --pretty Formatting

git log --pretty = format : “ format specifiers ”//use like output in C/C++

Format Specifiers:

%H

commit hash

%h

abbreviated commit hash

%T

tree hash

%t

abbreviated tree hash

%P

parent hashes

%p

abbreviated parent hashes

%an

author name

[Explore More Specifiers](#)

[Click here](#)

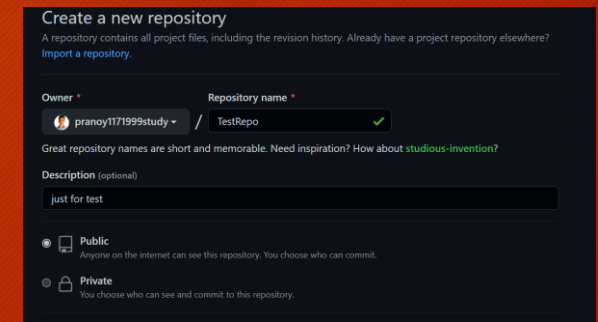
Example : “%H ~~~ %an” it will represents Commit Hash ~~~ Author Name

Unstaging and Unmodified Files

- `git restore --staged "file name"` (to Unstage a File)
- `git checkout - "file name"` (Unmodify File (Match to Last Commit))**Be Careful you will Loose Current File
- `git checkout -f` (Set Working Directory to the last Commit , all Files and we will loose our current edits)

Working with Remote Repositories

- Create a Repositories in GitHub
- Add your GitHub repo's Origin to your Local Project
 - git remote add origin <https://github.com/pranoy1171999study/TestRepo.git>
- To check Remote Name : git remote
- To see fetch & push url : git remote -v
- Now Select the Branch : git branch -M main



Now Can we Push our local directly to the GitHub repo? 🤖

Answer is No 😞. We have to give the access of GitHub Repo to your local Git. Actually this is good otherwise someone who know your origin can push anything in your repo 😊.

```
pranoy patra@LAPTOP-II8GV53Q MINGW64 /d/SwabhavTechlabs/testGit
$ git init
Initialized empty Git repository in D:/SwabhavTechlabs/testGit/.git/

pranoy patra@LAPTOP-II8GV53Q MINGW64 /d/SwabhavTechlabs/testGit (master)
$ git remote add origin https://github.com/pranoy1171999study/TestRepo.git

pranoy patra@LAPTOP-II8GV53Q MINGW64 /d/SwabhavTechlabs/testGit (master)
$ git remote -v
origin https://github.com/pranoy1171999study/TestRepo.git (fetch)
origin https://github.com/pranoy1171999study/TestRepo.git (push)

pranoy patra@LAPTOP-II8GV53Q MINGW64 /d/SwabhavTechlabs/testGit (master)
$ git branch -M main

pranoy patra@LAPTOP-II8GV53Q MINGW64 /d/SwabhavTechlabs/testGit (main)
$ git push -u origin main
error: src refspec main does not match any
error: failed to push some refs to 'https://github.com/pranoy1171999study/TestRepo.git'
```

Now What is the Solution Then? How to give Access ? 🤔

For this we have to goto GitHub and add our local Git's SSH Key in it

Step 1: Generate SSH key in Git Bash

->ssh-keygen -t ed25519 -C "your_email@example.com" (Generate SSH Key)

-> eval "\$(ssh-agent -s)" (Run SSH agent)

->ssh-add ~/.ssh/id_ed25519 (Add your SSH Private Key to SSH Agent)

->tail ~/.ssh/id_ed25519.pub (See your SSH Key)

Step 2 :Open GitHub & add new SSH Key(Local Git's SSH Key)(In the Pic Red portion)

GitHub Website ->Click on Profile Icon-> Settings->SSH & GPG Keys -> New SSH Key
give a title & paste the SSH Key & add it

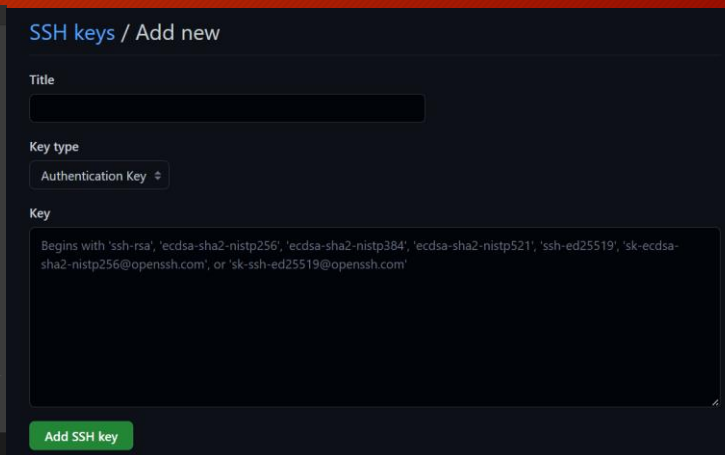
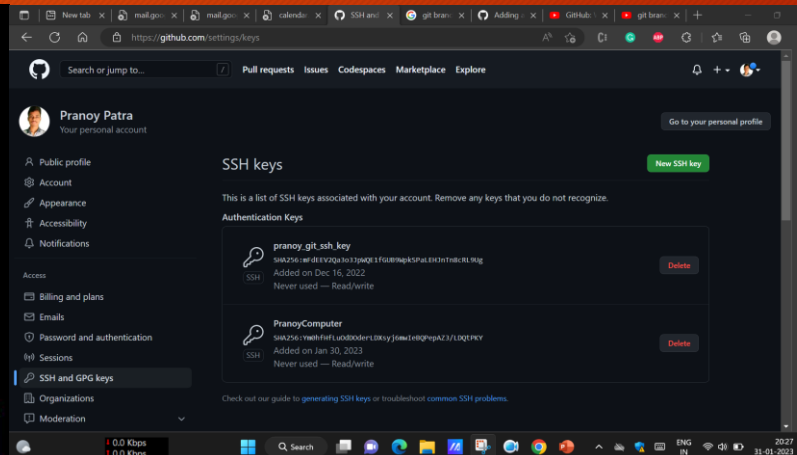
[Explore More](#)

```
pranoy patra@LAPTOP-II8GV53Q MINGW64 /d/SwabhavTechlabs/testGit (main)
$ ssh-keygen -t ed25519 -C pranoypatra1171999study@gmail.com
Generating public/private ed25519 key pair.
Enter file in which to save the key (/c/Users/pranoy patra/.ssh/id_ed25519):
/c/Users/pranoy patra/.ssh/id_ed25519 already exists.
Overwrite (y/n)?

pranoy patra@LAPTOP-II8GV53Q MINGW64 /d/SwabhavTechlabs/testGit (main)
$ eval "$(ssh-agent -s)"
Agent pid 892

pranoy patra@LAPTOP-II8GV53Q MINGW64 /d/SwabhavTechlabs/testGit (main)
$ ssh-add ~/.ssh/id_ed25519
Identity added: /c/Users/pranoy patra/.ssh/id_ed25519 (pranoypatra1171999study@gmail.com)

pranoy patra@LAPTOP-II8GV53Q MINGW64 /d/SwabhavTechlabs/testGit (main)
$ tail ~/.ssh/id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIGCJXkd43S+dz1f0ZI6dEVjml1Pv8QT7cUsyJkm66I+p
pranoypatra1171999study@gmail.com
```



Push Project to GitHub

Now go to GitBash and run the Commands

git commit (Commit your changes)

git push -u origin main (Push main Branch to origin)

😊 ★ Your Files are Uploaded Successful to GitHub 😊 ★

```
MINGW64:/d/SwabhavTechlabs/GitRepresentation
pranoy patra@LAPTOP-II8GV53Q MINGW64 /d/SwabhavTechlabs/GitRepresentation (main)
$ git add .

pranoy patra@LAPTOP-II8GV53Q MINGW64 /d/SwabhavTechlabs/GitRepresentation (main)
$ git commit -m "hii"
[main 476c057] hii
1 file changed, 0 insertions(+), 0 deletions(-)

pranoy patra@LAPTOP-II8GV53Q MINGW64 /d/SwabhavTechlabs/GitRepresentation (main)
$ git push -u origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 544.84 KiB | 11.35 MiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/pranoy1171999study/Git-GitHubPresentation.git
   a124a41..476c057  main -> main
branch 'main' set up to track 'origin/main'.

pranoy patra@LAPTOP-II8GV53Q MINGW64 /d/SwabhavTechlabs/GitRepresentation (main)
$ |
```


Git Alias

What is Git Alias? 🤖

Git Alias is nothing but a feature provided by Git to make your custom shortcuts. You can set some custom command for any Git Bash command.

Ex: `git config --global alias.st status`
(Now you can use “git st” instead of git “git status”)

```
MINGW64:/c/Users/pranoy patra/GitLearn
pranoy patra@LAPTOP-II8GV53Q MINGW64 ~/GitLearn (main)
$ git config --global alias.st status

pranoy patra@LAPTOP-II8GV53Q MINGW64 ~/GitLearn (main)
$ git st
On branch main
Your branch is up to date with 'origin/main'.

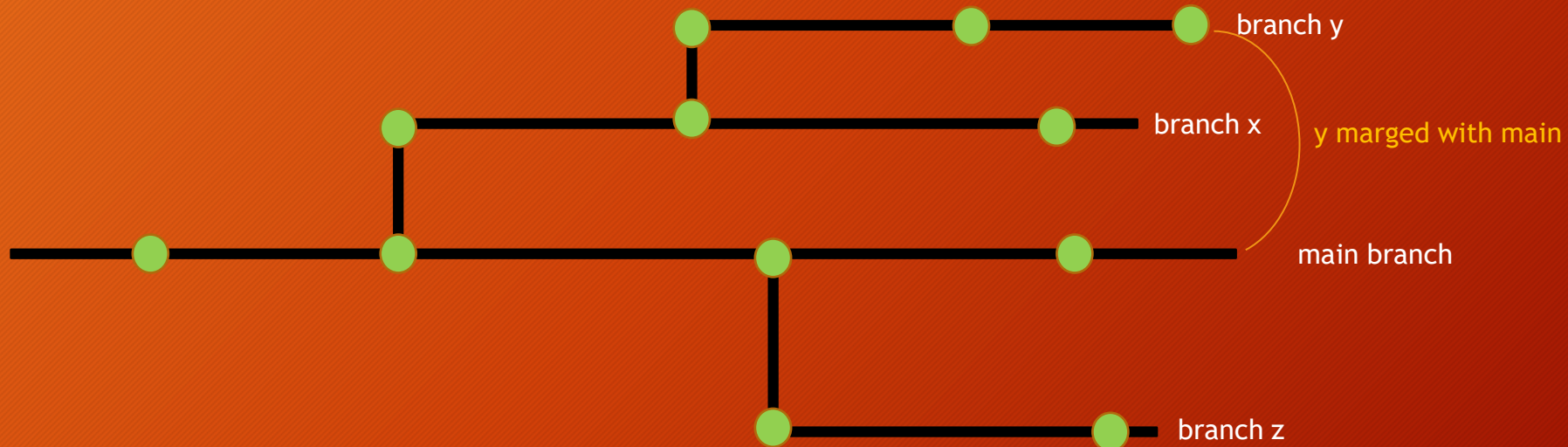
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
  (commit or discard the untracked or modified content in submodules)
        modified:   analog_clock (modified content)

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

pranoy patra@LAPTOP-II8GV53Q MINGW64 ~/GitLearn (main)
$ |
```

Branches in Git

You can create a branch any time from any branch & do different different tasks & testing on them , after that you also can marge this branch with main(Production Branch)



Create a Branch

git checkout -b “new branch name” (Create a branch with given name)

git branch (See all available branches)

git checkout “Branch name” (Go to any branch)

****when you will switch branches your folders and files will change in memory w.r.t your current branch**

```
pranoy patra@LAPTOP-II8GV53Q MINGW64 ~/GitLearn (main)
$ git checkout -b "newBranch"
Switched to a new branch 'newBranch'

pranoy patra@LAPTOP-II8GV53Q MINGW64 ~/GitLearn (newBranch)
$ git branch
  devlopedByPranoy
  main
* newBranch

pranoy patra@LAPTOP-II8GV53Q MINGW64 ~/GitLearn (newBranch)
$ git checkout devlopedByPranoy
Switched to branch 'devlopedByPranoy'
M    analog_clock

pranoy patra@LAPTOP-II8GV53Q MINGW64 ~/GitLearn (devlopedByPranoy)
$ git branch
* devlopedByPranoy
  main
  newBranch

pranoy patra@LAPTOP-II8GV53Q MINGW64 ~/GitLearn (devlopedByPranoy)
$ |
```