**GIT-NOTES-EASY-WAY**

Definition: -😍️ (software code store)

**Git** is an **open-source distributed version control system**. It is designed to handle minor to major projects with high speed and efficiency. It is developed to co-ordinate the work among the developers. The version control allows us to track and work together with our team members at the same workspace.

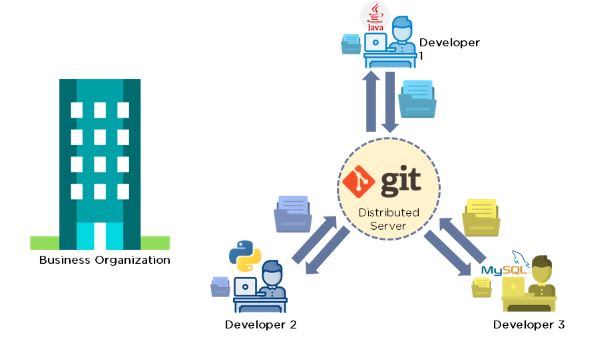
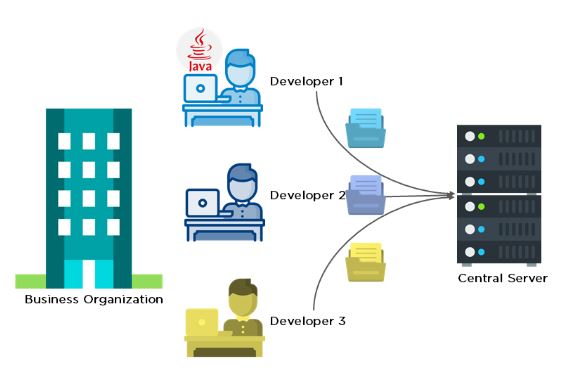
* Git is foundation of many services like **GitHub** and **GitLab**, but we can use Git without using any other Git services. Git can be used **privately** and **publicly**.
* Git was created by **Linus Torvalds** in **2005** to develop Linux Kernel. It is also used as an important distributed version-control tool for **the DevOps**.
* Git is easy to learn, and has fast performance. It is superior to other SCM tools like Subversion, CVS, Perforce, and ClearCase.

**Or**

**Git** is a DevOps tool used for source code management. It is a free and open-source version control system used to handle small to very large projects efficiently. Git is used to tracking changes in the source code, enabling multiple developers to work together on non-linear development.

**or**

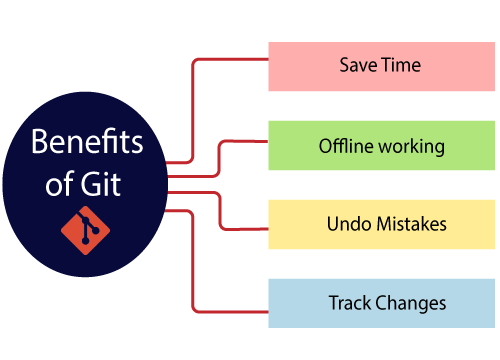
**Git** is a version control system used for tracking changes in computer files. It is generally used for source code management in software development. Git is used to tracking changes in the source code. The distributed version control tool is used for source code management



**Features of git**

* Tracks history
* Free and open source
* Supports non-linear development
* Creates backups
* Scalable
* Supports collaboration
* Branching is easier
* Distributed development

**Benefits of Git: -**



**Git setup**--------------------------1 😍️

Install and setup Git & Git hub (Local and remote)

* <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git> (Windows)
* sudo apt update (Linux)
* sudo apt install git
* git –version
* Output
* git version 2.34.1

**Create GitHub in remote………….2** 😍️

* <https://github.com/signup?ref_cta=Sign+up&ref_loc=header+logged+out&ref_page=%2F&source=header-home>
* Setup GitHub account

add the GitHub config to local git (config means working with user git config)

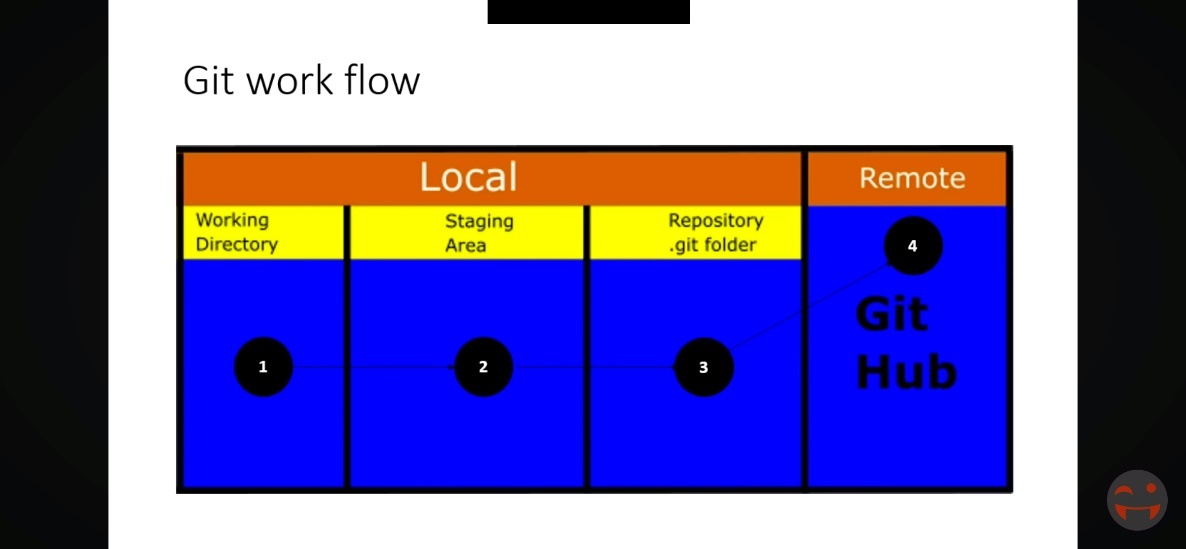
**global.... (same as in remote server account details in local)**

* git –version
* git config --global -e (edit)
* git config --global user.name "MaheshBabu32"
* git config --global user.email [3205mahi@gmail.com](mailto:3205mahi@gmail.com)
* git config –list
* git help <command name>

**setup-3...............(setup notepad++ easy editor)** 😍️

* sudo apt update && sudo apt upgrade -y
* snap
* sudo snap install notepad-plus-plus
* notepad-plus-plus
* ls
* notepad++ .bash\_profile
* notepad++
* notepad-plus-plus
* notepad-plus-plus .bash profile

**Git-Diagram-------------4** 😍️



**Git Workflow**-----------------------4😍️

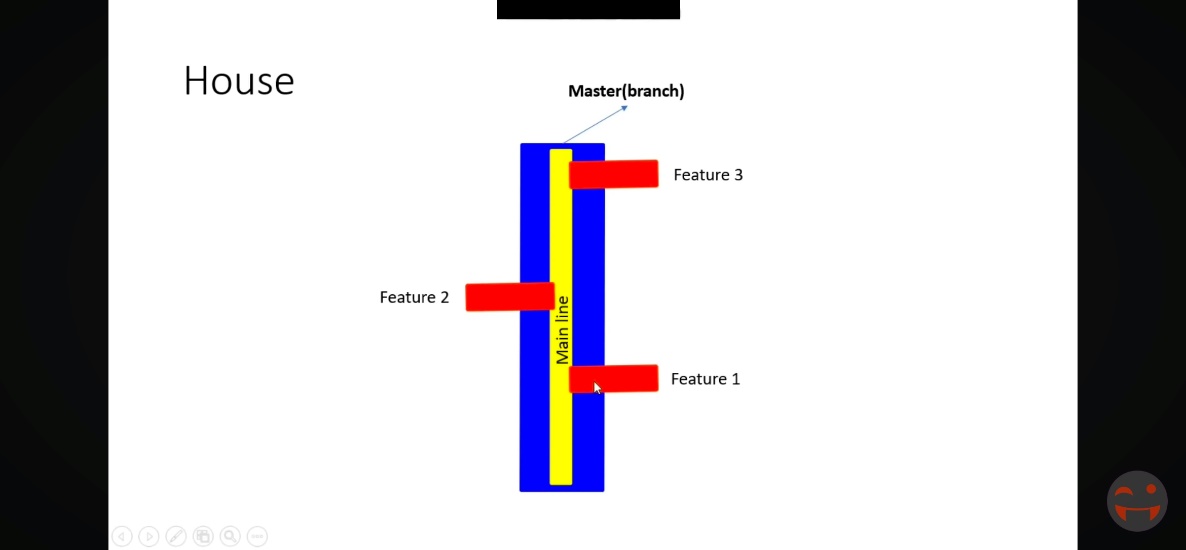
**Terms: -**

* **Commit** (who changes the code it will show the date and time (Author))
* **Clone** (another name is copy eg: - In github code is their same code clone to the local git)
* **Tracking & Untracking** (. git means it will track the project)
* **Repository** (it means space or directory (Either local, Github, or any online host)

or

A repository is the most basic element of GitHub. It's a place where you can store your code, your files, and each file's revision history. Repositories can have multiple collaborators and can be either public or private.

**Git(process)...................5**😍️



1. **Master--Feature branch (Additional)**
2. **Origin**

**Eg:- HOME**

* Default--**Master** (Somewhere the project is clone into the local repo the default branch name is master)
* Based on requirement create additional branch called feature branch (Feature 1, 2 3 4 5)

**Eg:- speaker someone info**

Default—

* **Origin** (The project is Cloning from remote servers like GitHUb, GitLab, into local git so the default name is ORIGIN (Server name or project name)

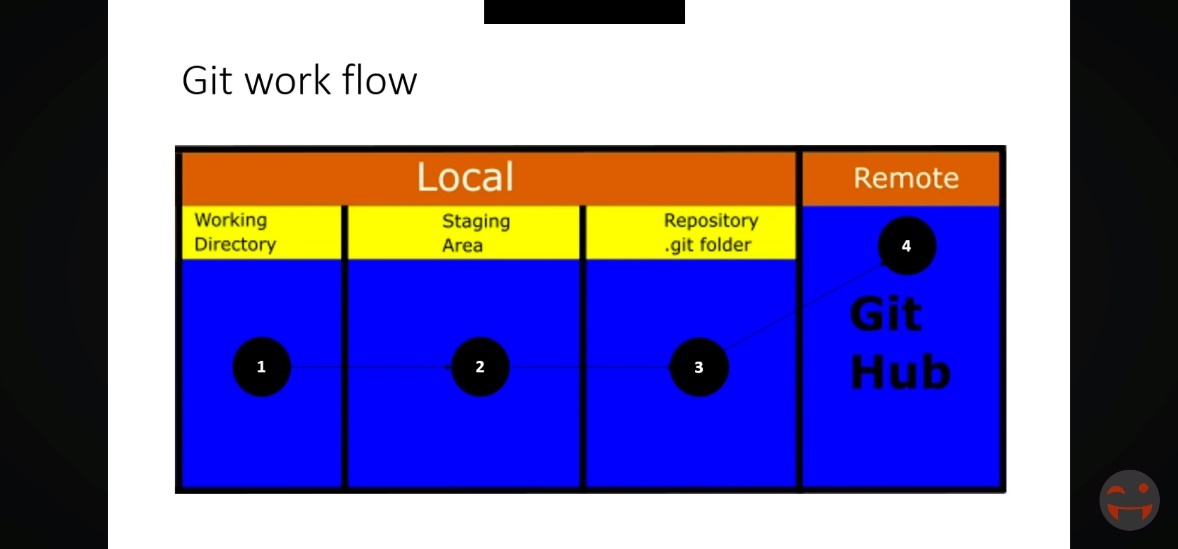
**Git stages: -😍️-SRC-code**

1. Working Directory (git add) ---file(home.html)

2. Staging area (git commit) ----stage to local repo (messege)

3. LocalRepo (git push) ---code in local repo

4. RemoteRepo (git pull)



**How to push the src code from local to remote (Git to github, or GitLab, or Tuleap**

**They are two methods: - (https, or shh)-protocols**

* **Fork process** (person A code I clone into my GitHub account (B) After that code we have to copy in local git if any edit or any changes that code after again push into our git hub (B). -------------**method 1 https**

**Steps: -**

* Open terminal gitbash or Linux terminal
* Mkdir project\*\*
* Cd project/\*\*
* A person code im clone into my B person account
* Git clone <https://github.com/MaheshBabu32/suresh-techs.git> <new name>\*\*\*
* Ls\*\*
* Any code changes we have to change in vscode or notepad++
* ls
* git status\*\*
* ls -la (check. git folder is tracking our project)
* git add <project name>\*\*
* git commit -m <message>\*\*
* git config --global user.name "MaheshBabu32" \*\*
* git config --global user. email [3205mahi@gmail.com](mailto:3205mahi@gmail.com) \*\*
* git config –global -e
* git log -v (commit mess status) \*\*\*
* git push -u origin master \*\*\*\*

The code it will go to remote server

**Method-1.2 is developing our own code in local repo so that code we have push into to the remote server (https)**

* Create an account GitHub in remote server after that create one new repository
* Come to local git
* mkdir project\*\*\*
* cd project/\*\*\*
* touch f1 \*\*\*
* vi f1 (java, python, c)\*\*\*
* save and exit
* ls
* git status\*\*
* git init (.git)\*\*
* git status\*\*
* git branch --all
* ls -al\*\*
* git config --global user.name "-----------" \*\*\*
* git config --global user. email "-------------" \*\*\*
* git config –global -e \*\*\*
* git config –list
* git restore –staged <filename> or **.**\*\*\*\*\* (it will go to unstaged)
* git add <file name>\*\*\*\*
* git commit -m <message>\*\*\*
* git remote add origin <https://github.com/MaheshBabu320/first.git> \*\*\*\*\*\*\*
* git push -u origin master \*\*\*\*\*

The code will go into a remote server called (GitHub, Tuleap, Gitlab)

**In case any errors to push the code**

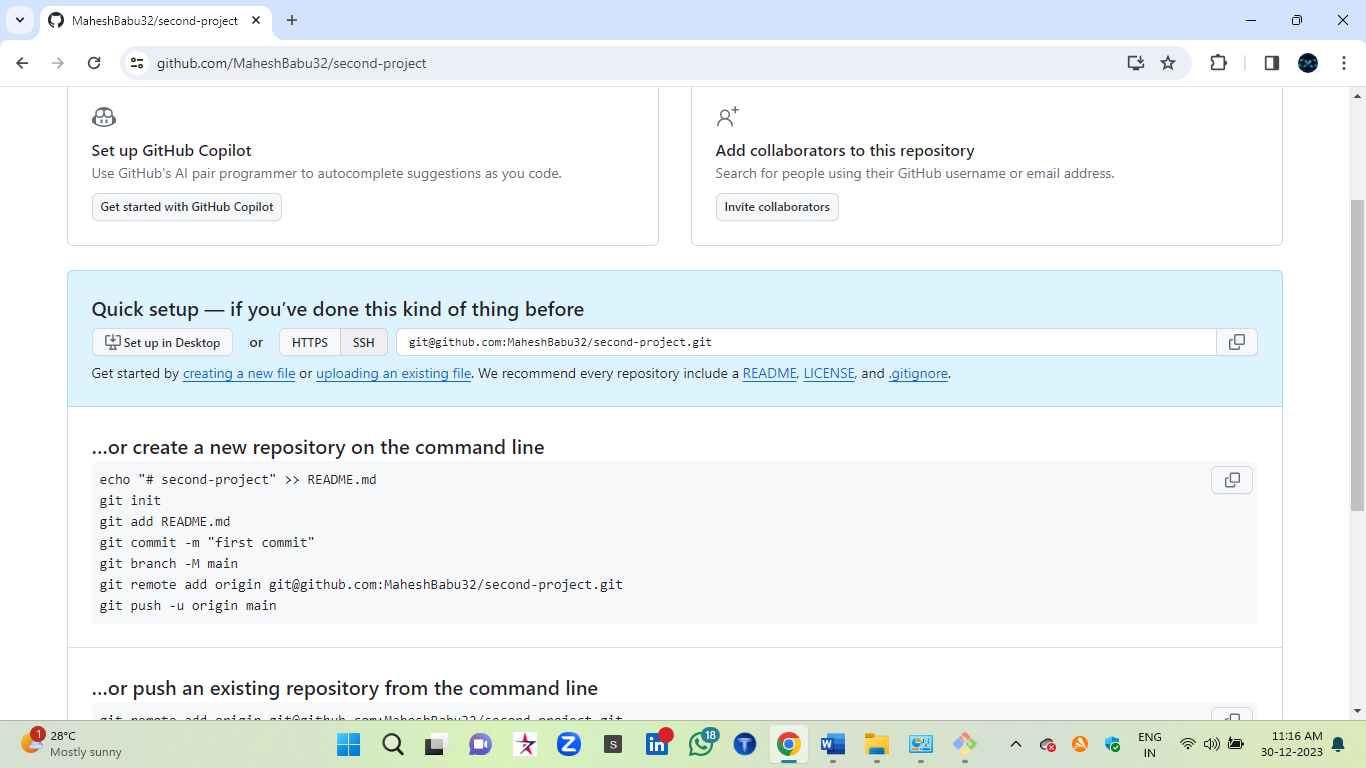
* in windows open **credential Manager** and remove old one account checks new one is update or not
* create token password in GitHub and add in local git

or

* ls -al ~/.ssh (check ssh key)
* ssh-keygen -t rsa -b 4096 -C [your\_email@example.com](mailto:your_email@example.com) (generate ssh key)
* eval "$(ssh-agent -s)" (add ssh key to agent)
* ssh-add ~/.ssh/id\_rsa (
* cat ~/.ssh/id\_rsa.pub (add ssh key to github)
* ssh -T [git@github.com](mailto:git@github.com) (verify ssh connection)
* git remote set-url origin [git@github.com:username/repository.git](mailto:git@github.com:username/repository.git) (update url)
* git push -u origin main (push to remote)

…………………………………HTTPS……………………………

**Method-2 (**By using **ssh** protocol to push the code from local to remote server

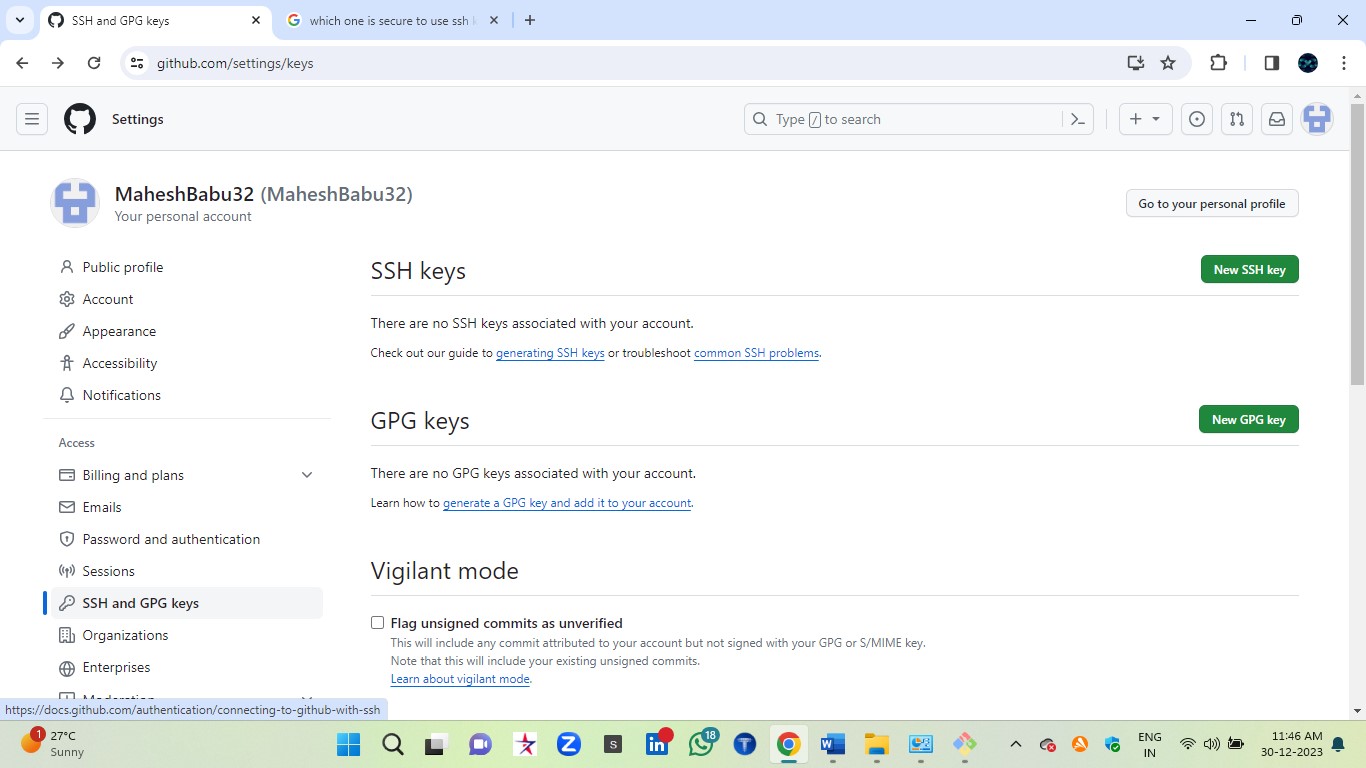


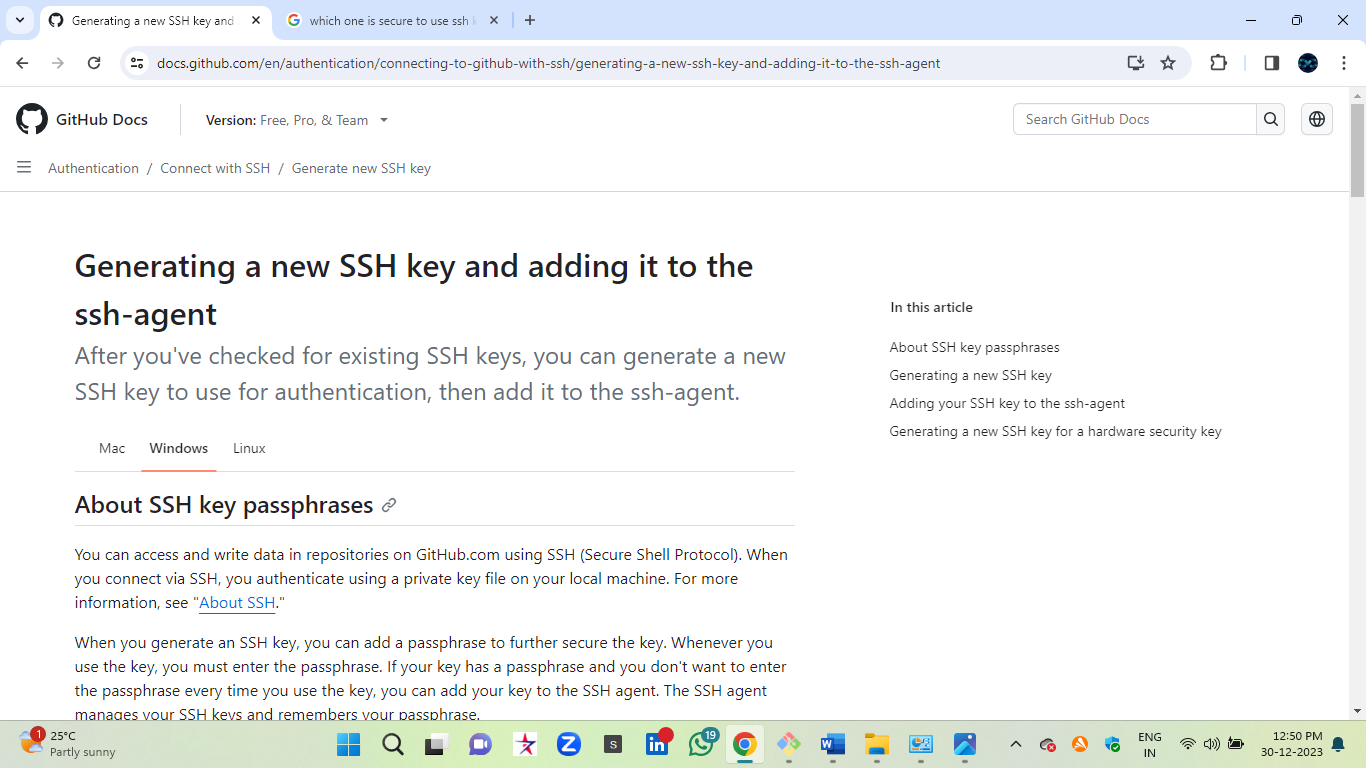
* By using **https** process (unless you go through a process of configuration anytime)-errors more.
* By using **SSH** 99.9% Realtime companies using ssh process and once connect to remote server no need to setup for next time (It will access directly our GitHub token).

**Processes\_ssh**

* Open terminal gitbash or Linux terminal
* Mkdir project\* (Create one empty folder)
* Cd project/\*
* Touch <f1>\*
* Vi <f1> (content)\*
* Ls
* Git init (initialize empty git repository)-tracking the files\*
* Git status \*
* Git add <filename>\*
* Git status (changes to be committed)\*
* Git config --global user.name “…………….”\*
* Git config --global user. email “………………….”\*
* Git config --global -e\*
* Git commit -m “message>\*
* Git log -v (check commit status)
* Git branch (check current branch name)
* Git branch -M main\*
* Git branch <name> (if u want to create new branch)
* Git checkout <branch name> (one branch to another branch shift)
* Git remote -v (this project is which repo is connected)
* git remote add origin [git@github.com:MaheshBabu32/second-project.git](mailto:git@github.com:MaheshBabu32/second-project.git) \*\*\*\*\*
* git push -u origin main\*\*\*
* So, we have to setup ssh key
* <https://docs.github.com/en/authentication/connecting-to-github-with-ssh>\*\*\*\*\*\*
* Click above link
* eval "$(ssh-agent -s)"\*\*\*\*
* ssh-keygen (two times enter)\*\*\*
* after created ssh key
* cd ~/.ssh/ (go inside ssh directory)\*\*\*\*\*\*
* ls\*\*\*\*
* id priv and id pub
* ssh-add id\_ed25519 (add to ssh agent by using private key)\*\*\*\*\*
* cat id\_ed25519.pub\*\*\*\*\*
* After this pub key add to git hub
* Github>settings>ssh and gpg keys>new ssh key>copy and paste>
* Final go to local git and go to our project dir
* git push origin main\*\*\*\*

The code will go into a remote server without any error!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!





……………………………………….SSH……………………………………….

So up to above process just we did only local git to remote GitHub and remote to local SRC code push and clone the code.! !!!!!!!!!!!!!!!!!!!!!!!!!!1

**Git commands\*\*\*\*\***

* sudo apt-get install git
* git –version

**Git Configuration & Setup**

* git config –global user.name “Your Name”
* git config –global user.email [youremail@example.com](mailto:youremail@example.com)
* git help
* man <command>

**Initializing a Repository**

* git init
* git init <directory>
* git clone <repository\_url>
* git clone –branch <branch\_name> <repository\_url>

**Basic Git Commands**

* git add <file>
* git add. or git add –all
* git status
* git status –ignored
* git diff
* git diff <commit1> <commit2>
* git diff –staged or git diff –cached
* git diff HEAD
* git commit
* git commit -m “<message>” or git commit –message “<message>”
* git commit -a or git commit –all
* git notes add
* git restore <file>
* git reset <commit>
* git reset –soft <commit>
* git reset –hard <commit>
* git mv

**Branching and Merging**

* git branch
* git branch <branch-name>
* git branch -d <branch-name>
* git branch -a
* git branch -r
* git checkout <branch-name>
* git checkout -b <new-branch-name>
* git checkout — <file>
* git merge <branch>
* git log
* git log <branch-d
* git log –all
* git stash
* git stash list
* git stash pop
* git tag
* git tag <tag-name>
* git tag <tag-name> <commit>
* git tag -a <tag-name> -m “<message>”

Remote Repositories

* git fetch
* git fetch <remote>
* git fetch –prune
* git pull
* git pull <remote>
* git pull –rebase
* git push
* git push <remote>
* git push <remote> <branch>
* git push –all
* git remote
* git remote add <name> <url>

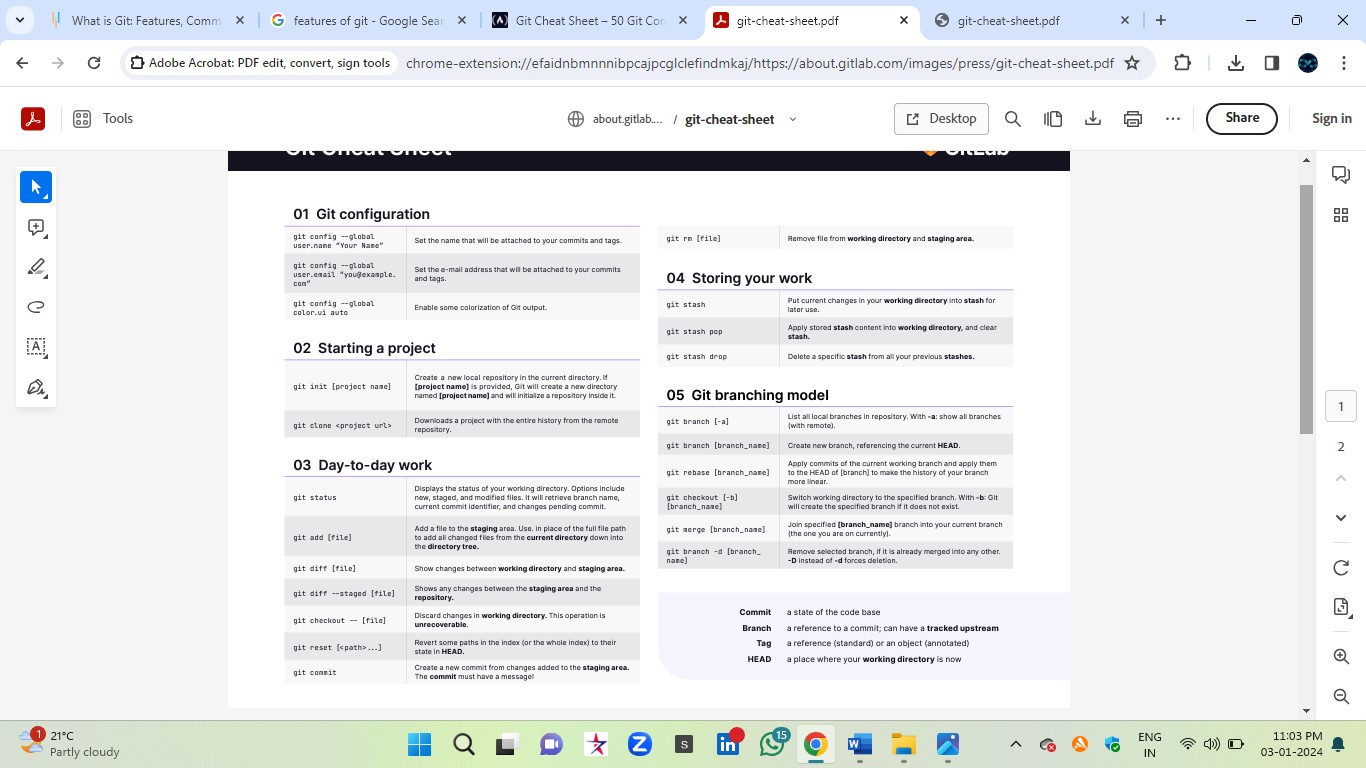
Git Comparison

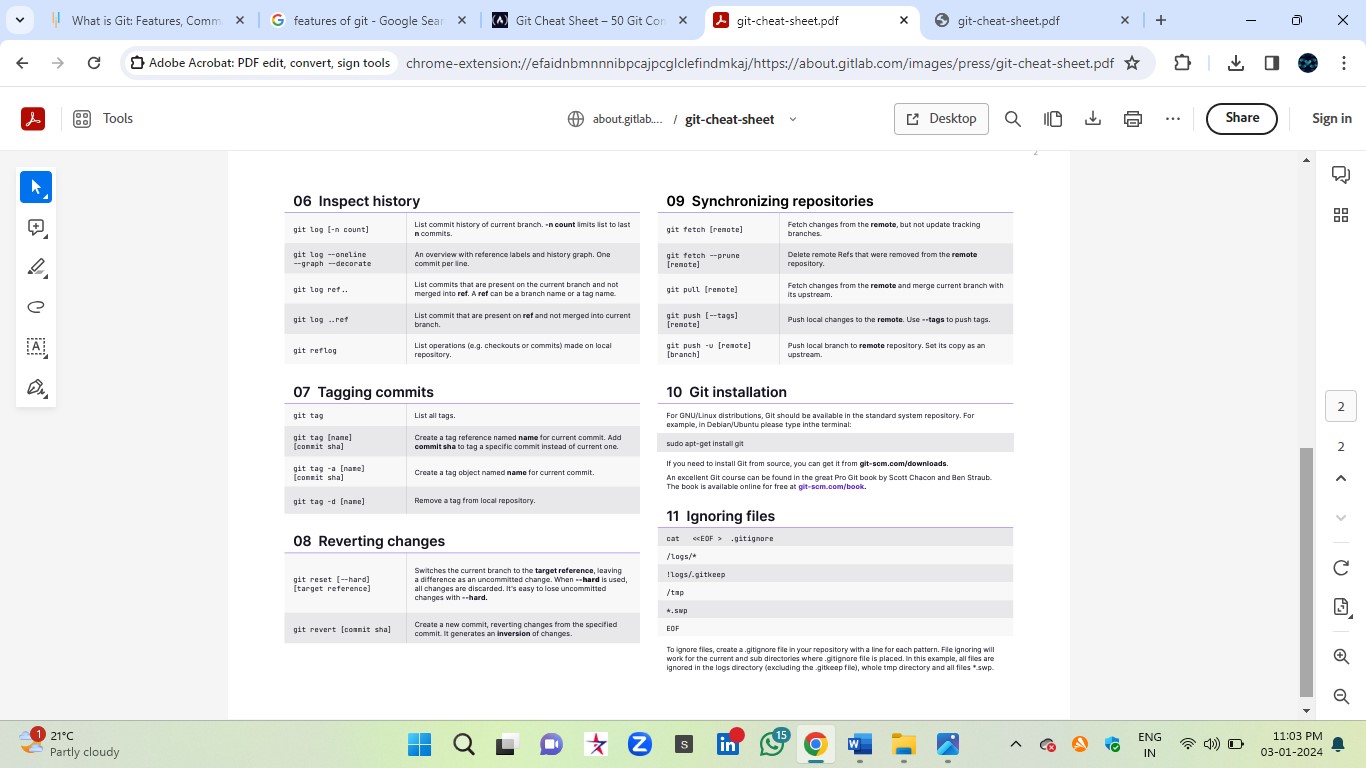
* git show
* git show <commit>

Git commands online <URL>

* <https://www.geeksforgeeks.org/git-cheat-sheet/> \*\*\*\*\*
* <https://www.freecodecamp.org/news/git-cheat-sheet/>

**Git commands sheet**





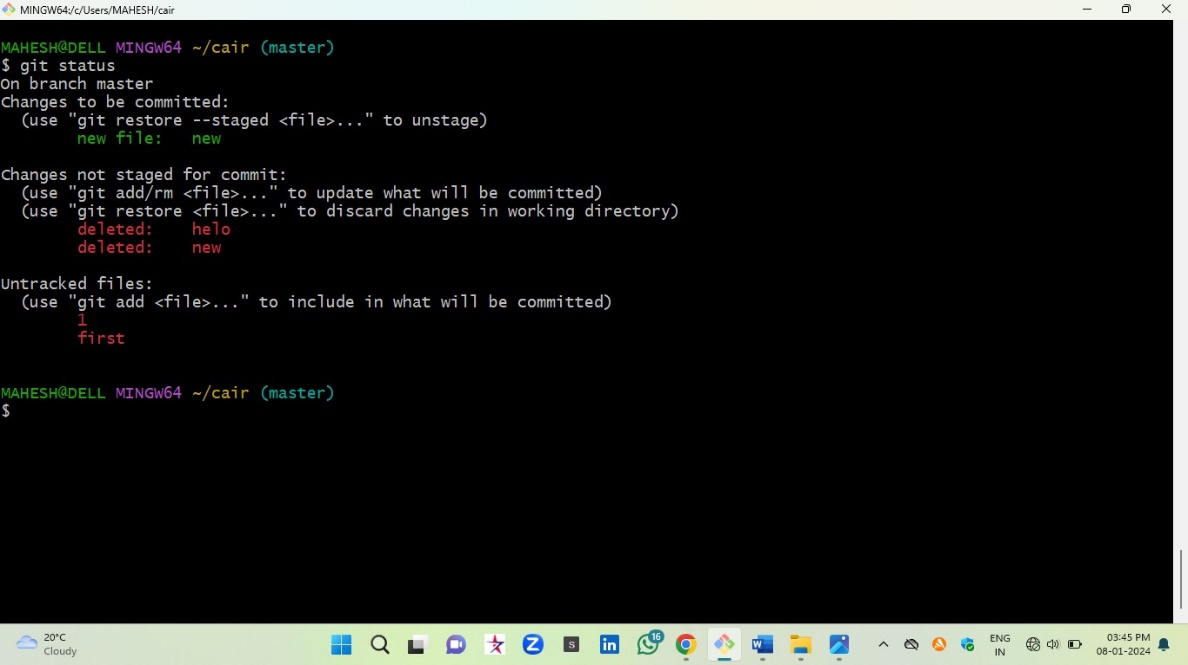
…………………………………………..Commands…………………………………………….

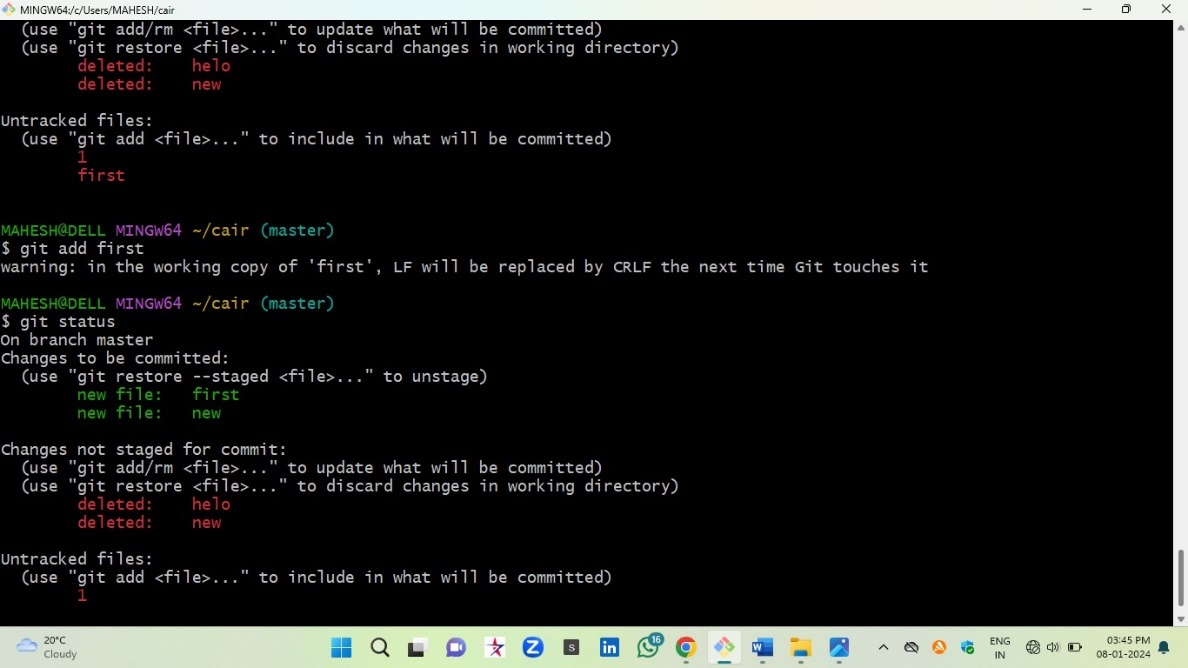
**Topic 6………………………..**

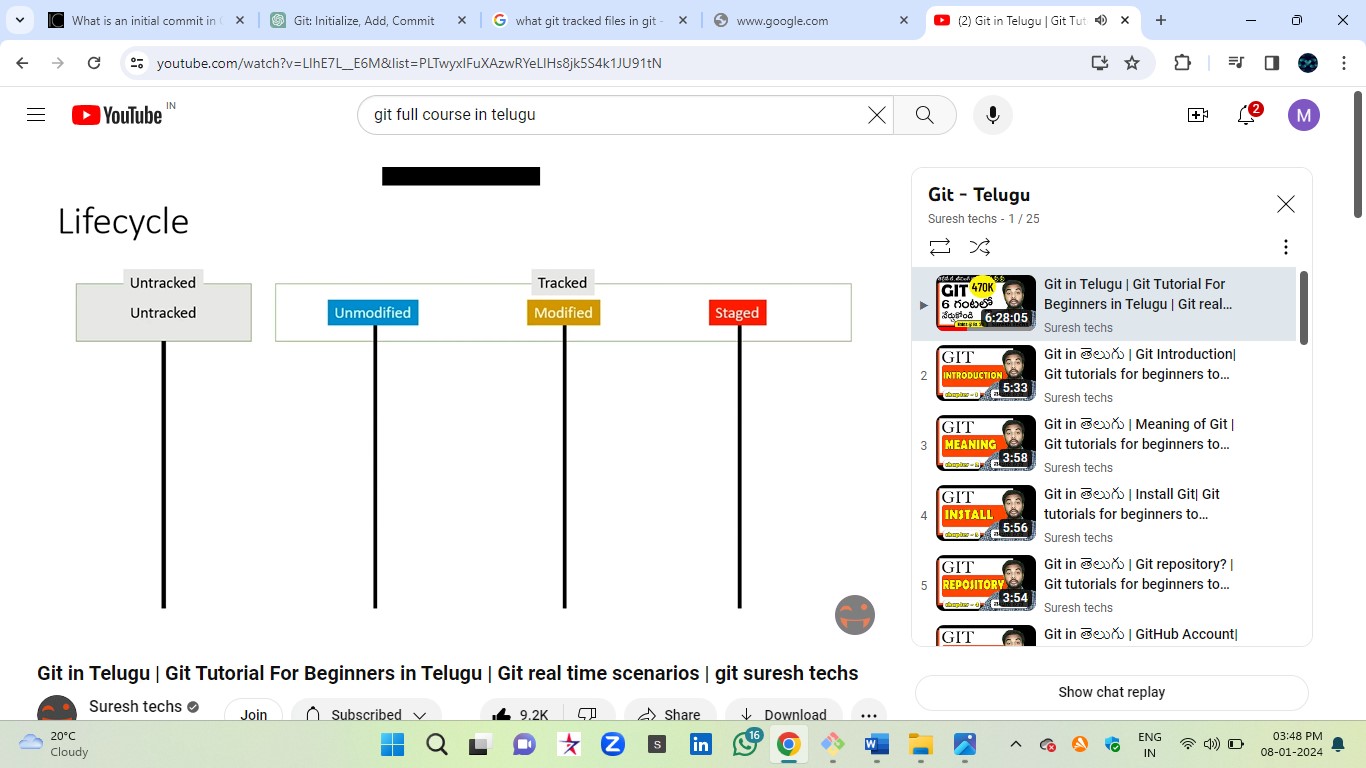
**Git status life cycle: -**

Basically, there are two types of files system in git

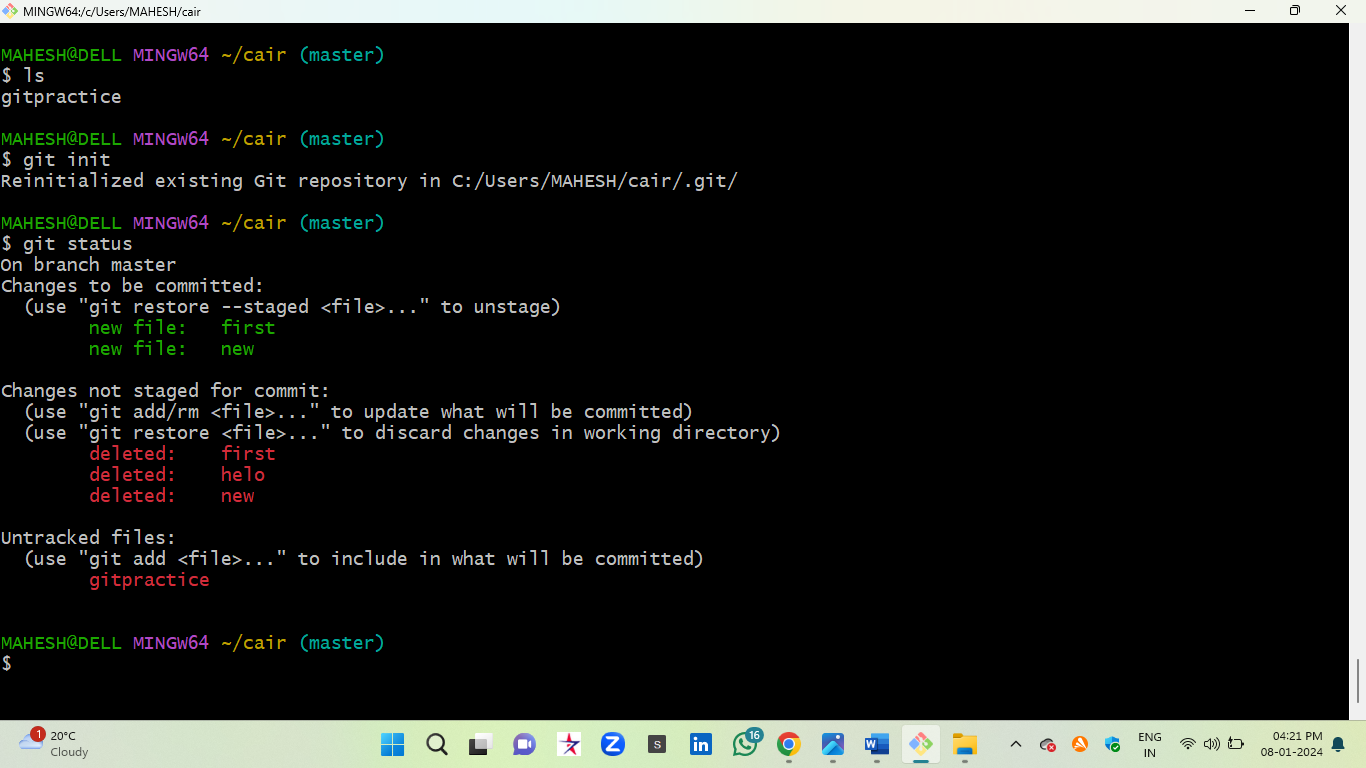
* Untracked files (git don’t know files) which hasn't yet been staged
* Tracked files (git know files) Changes not staged for commit
* Tracked files again 3 types





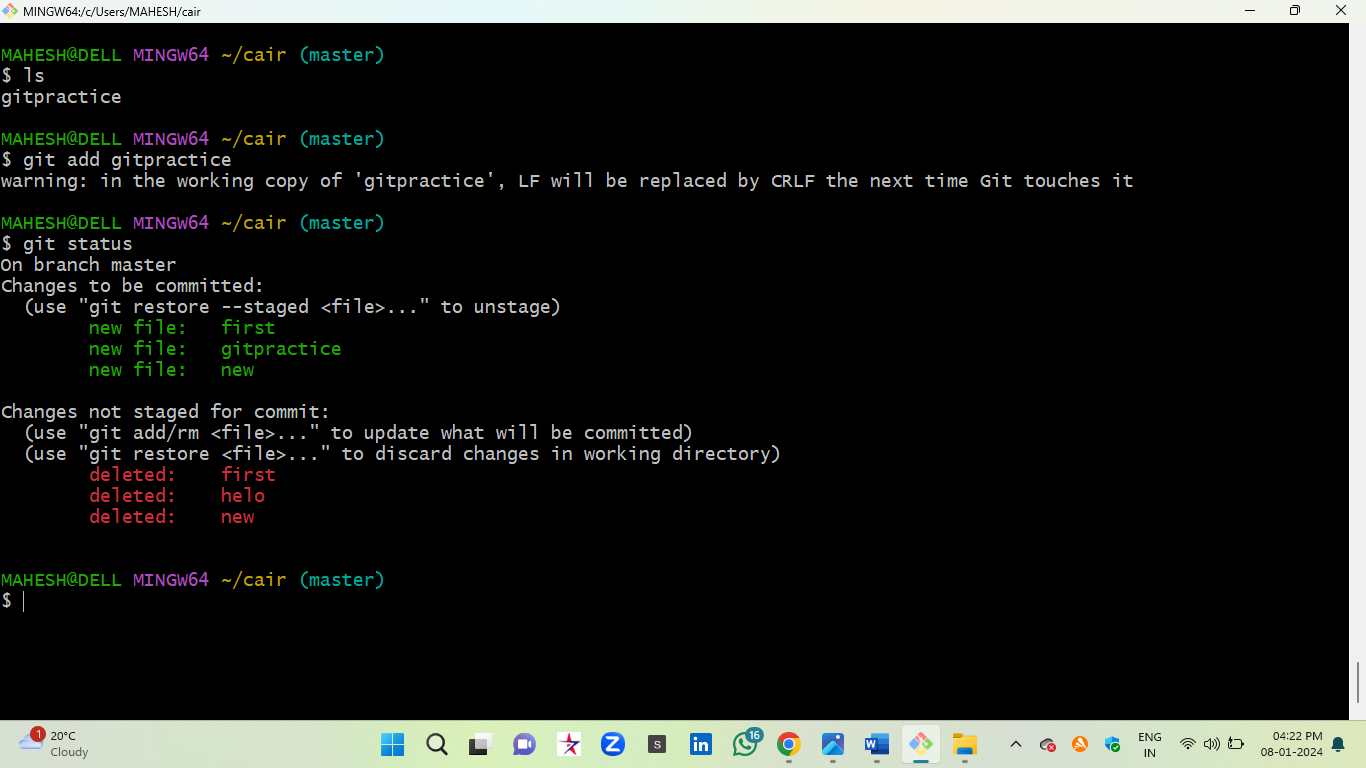


Untracked stage: -



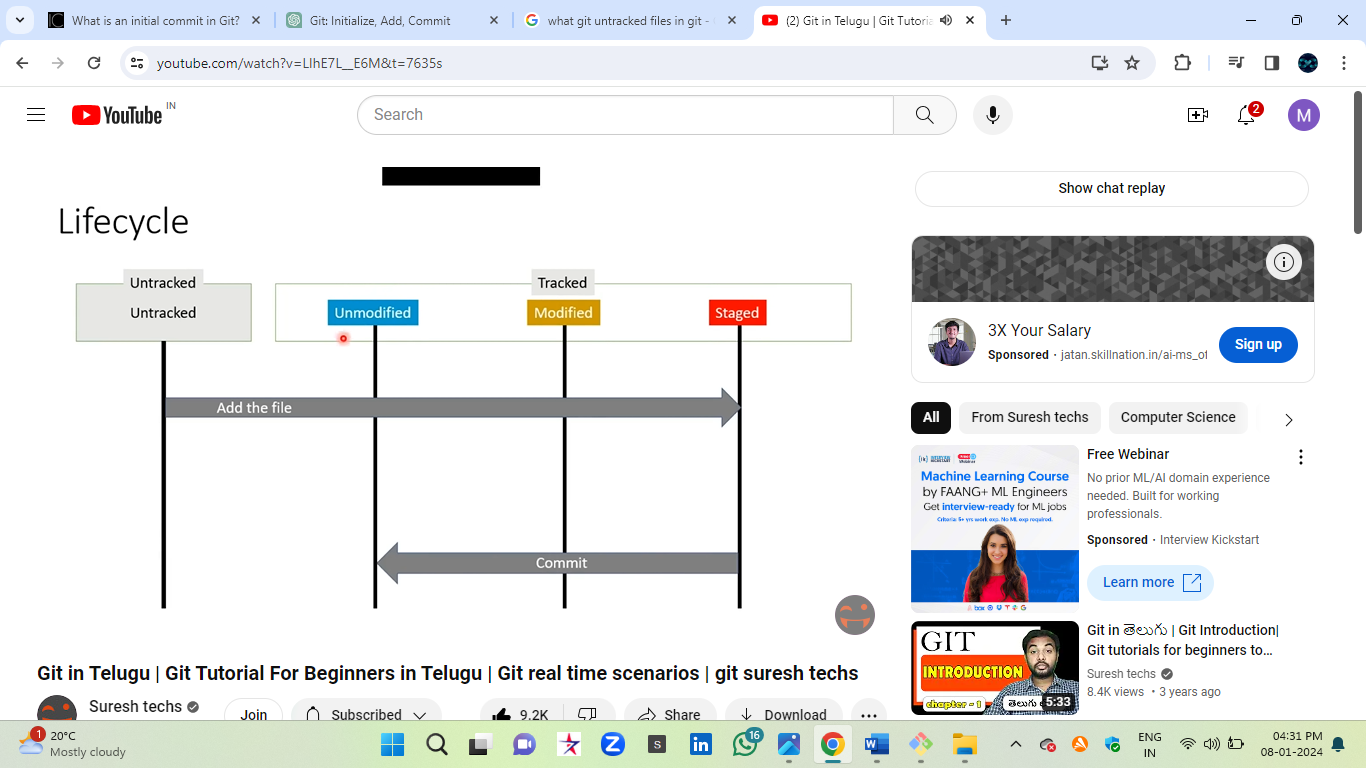
**Tracked stages: -**

stage 1 (or unstage) after adds the file

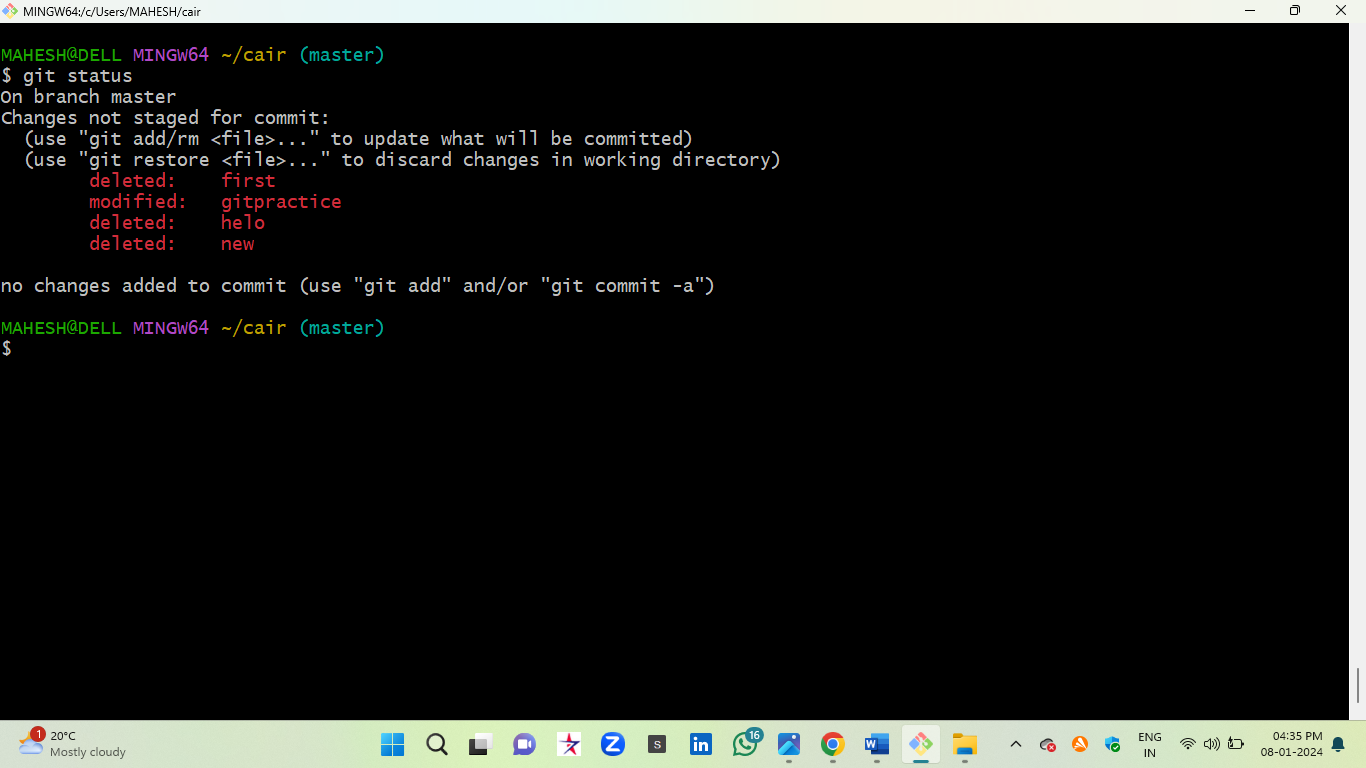




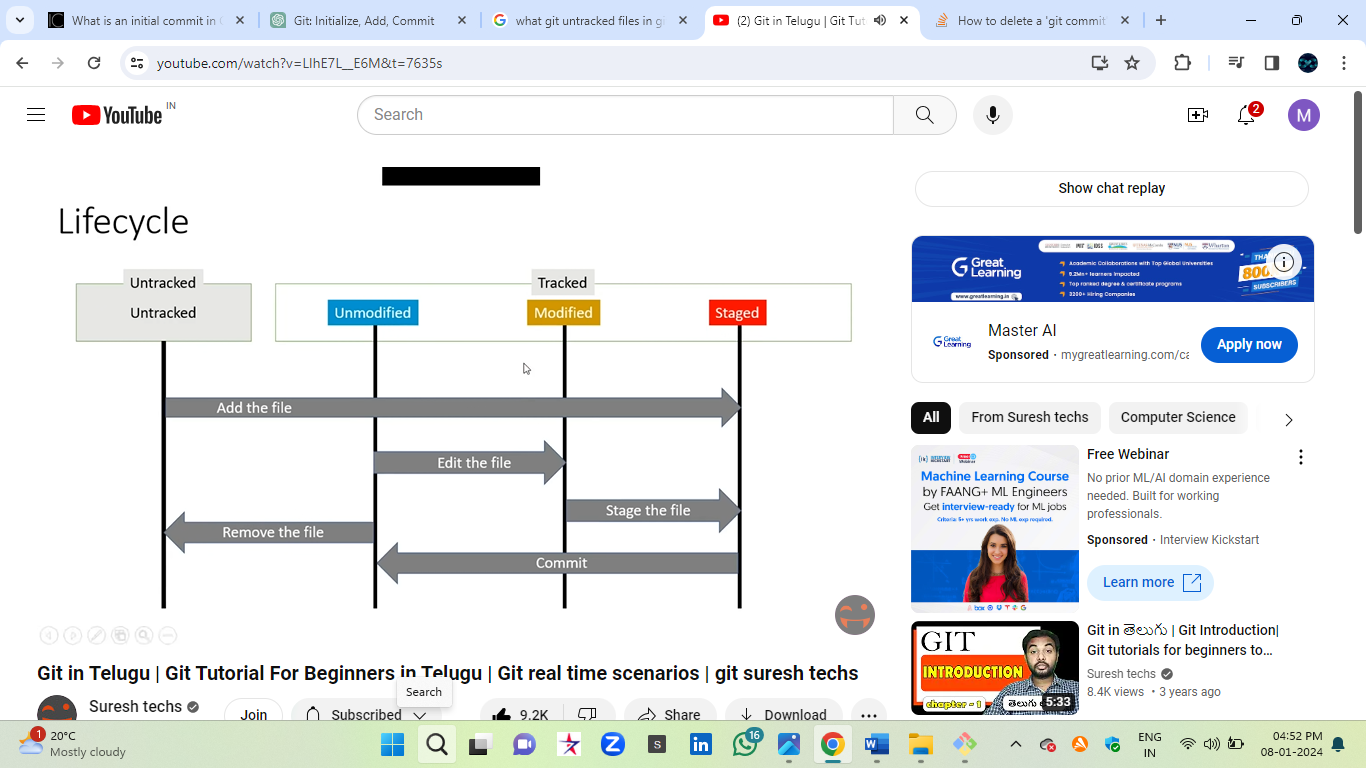
Unmodified stage: - (ready to push the code into remote)—After commit stage



Modified stage: - (edit stage)



* git reset --hard HEAD^ (remove git commit history data)



…………………………………..Git Status Life Cycle…………………………………………

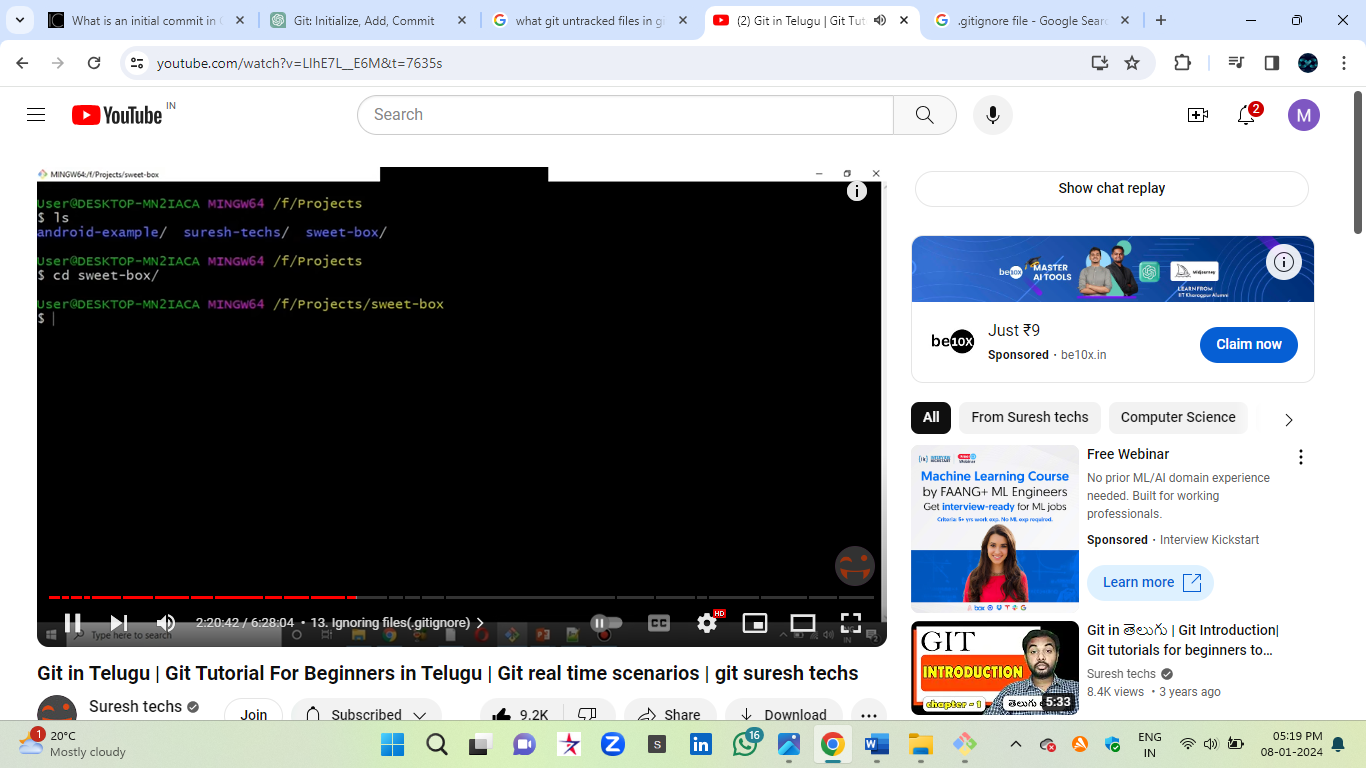
Step7

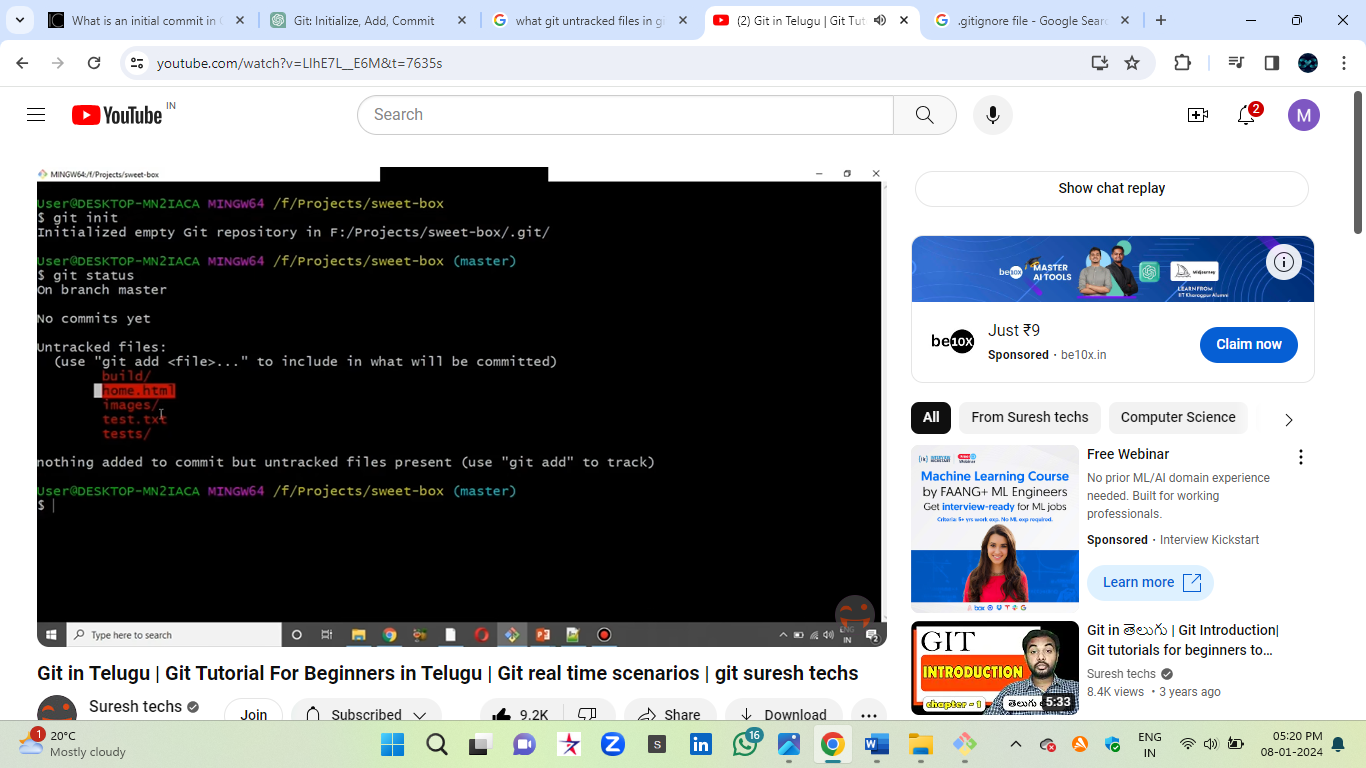
Ignoring files in git: -

Def: - Git can only ignore untracked files that haven't yet been committed to the repository.

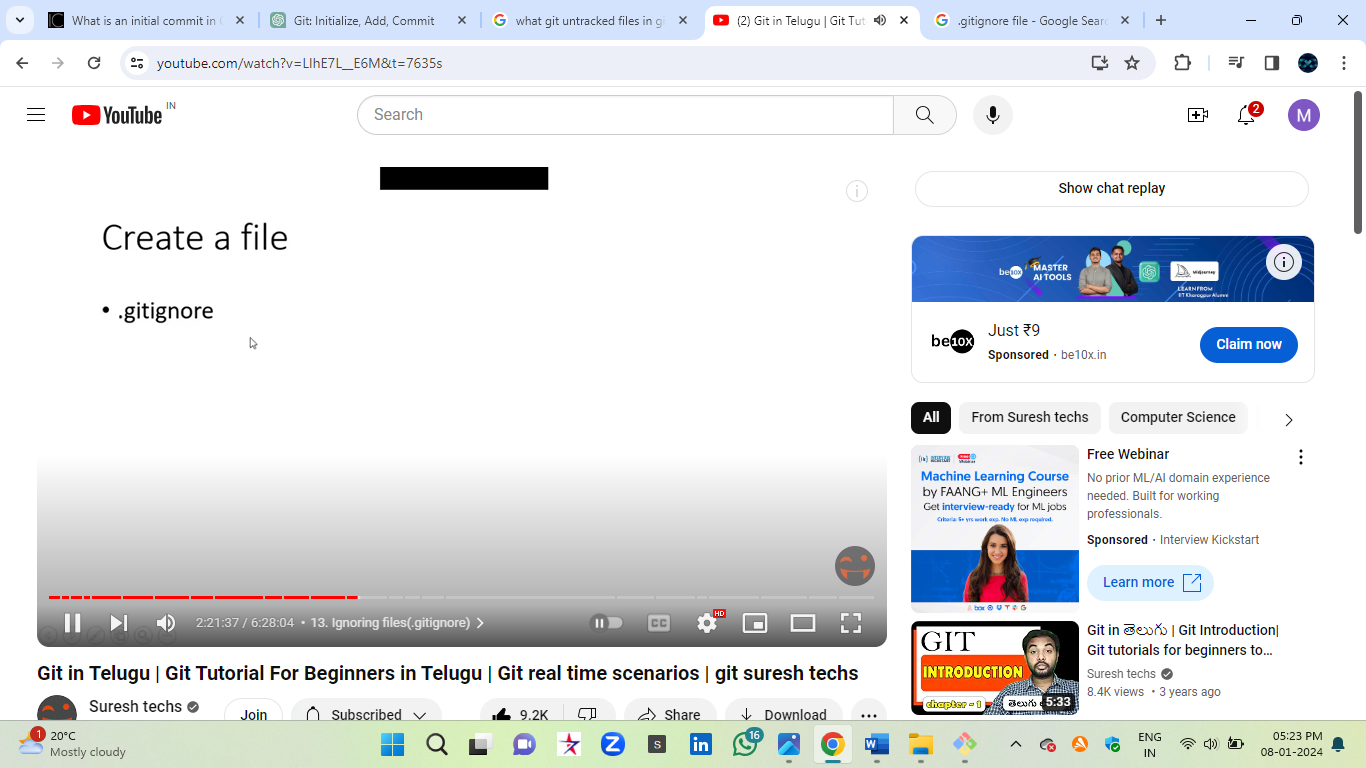
Eg: - I created f1 and f2 files, but I need to push the data only f1 not f2, so that time don’t delete the files, just simple ignore or hide the file( git is not tracked that file).

Steps:-

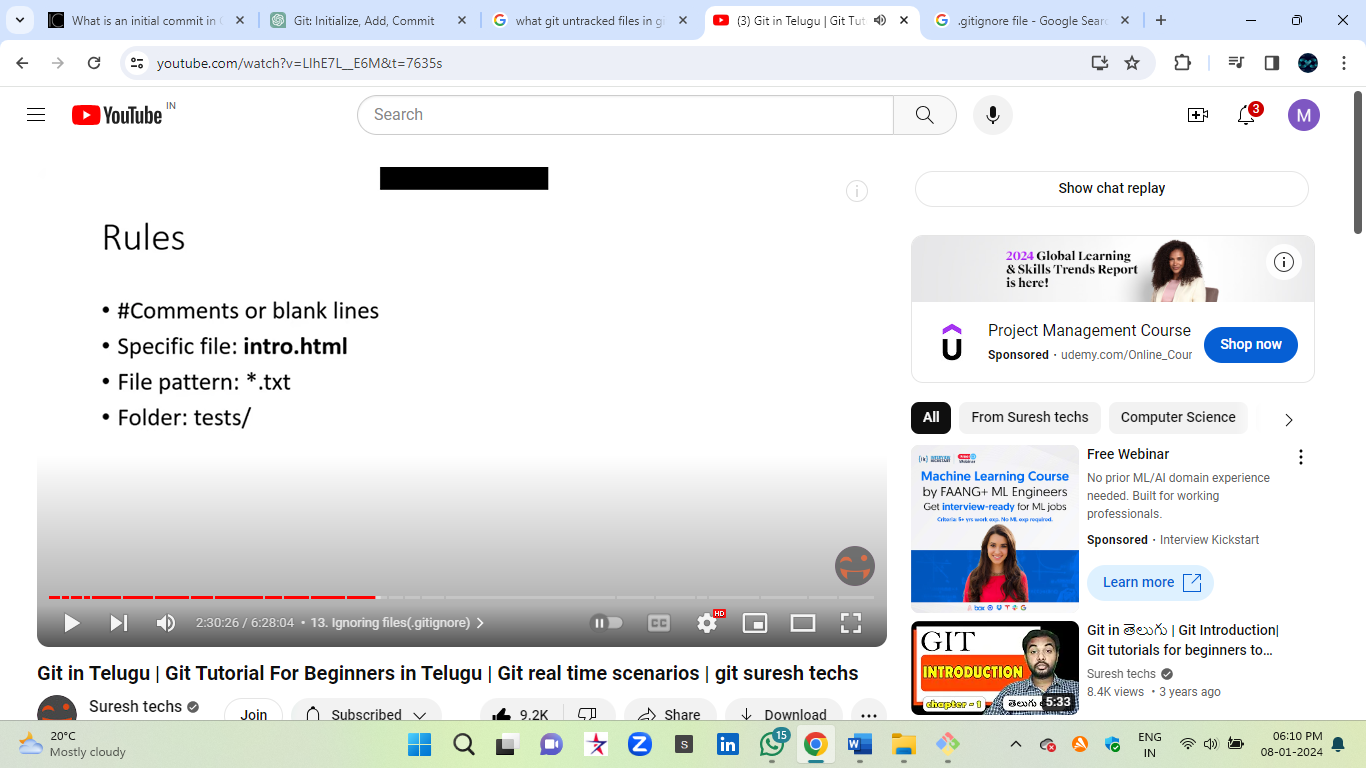




Create a .gitignore file



* vi .gitignore
* #comments can be placed…
* <filename> hide
* Ls
* Git status
* Now the file will hide



…………………………………….…..gitignore……………………………………………….

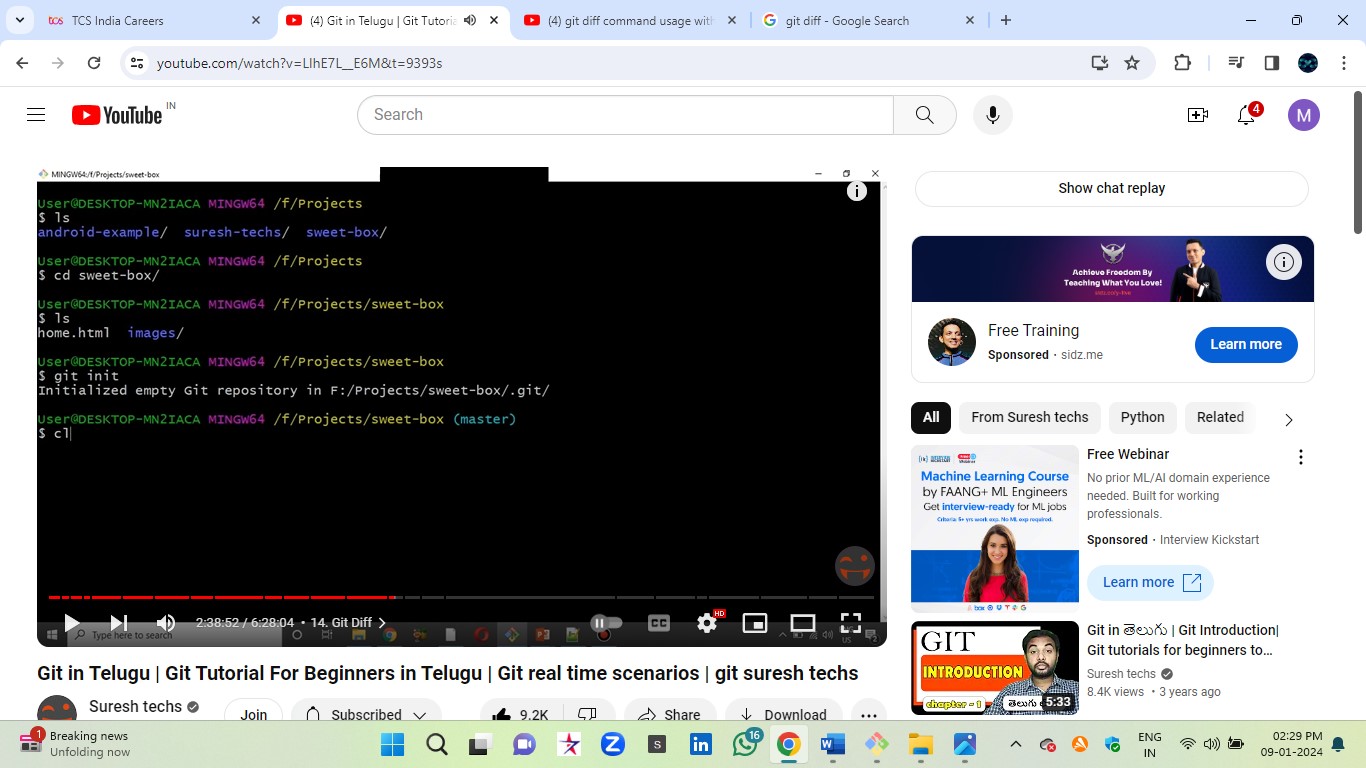
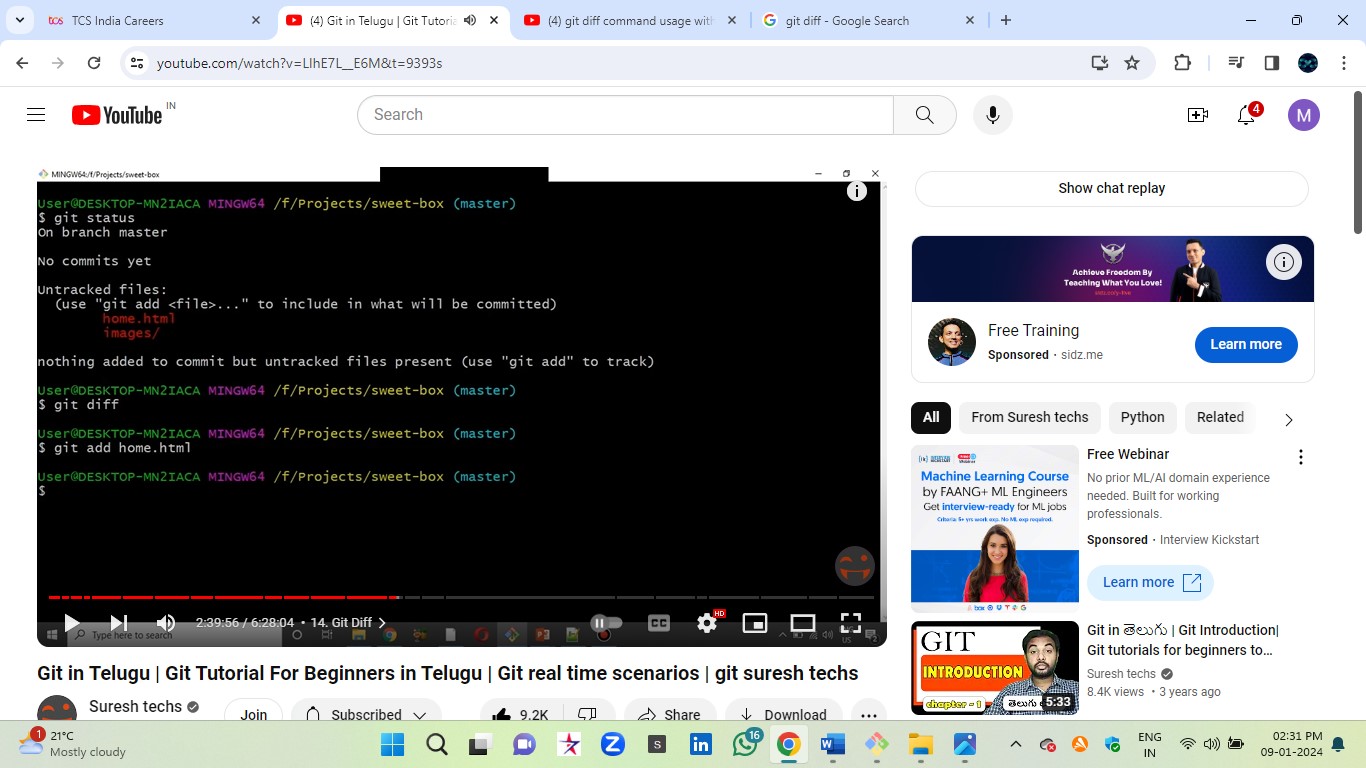
Step8---git diff

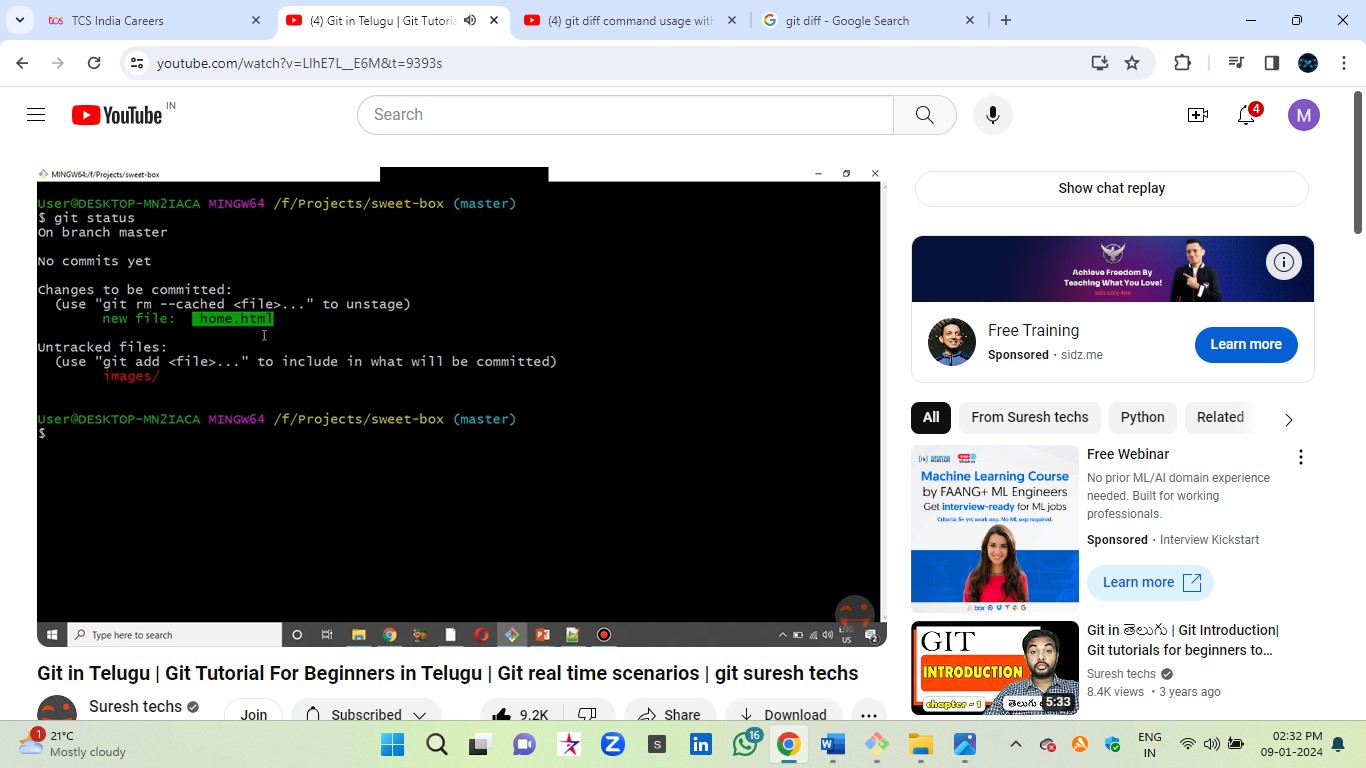
Diff:- The diff between working directory and staging area

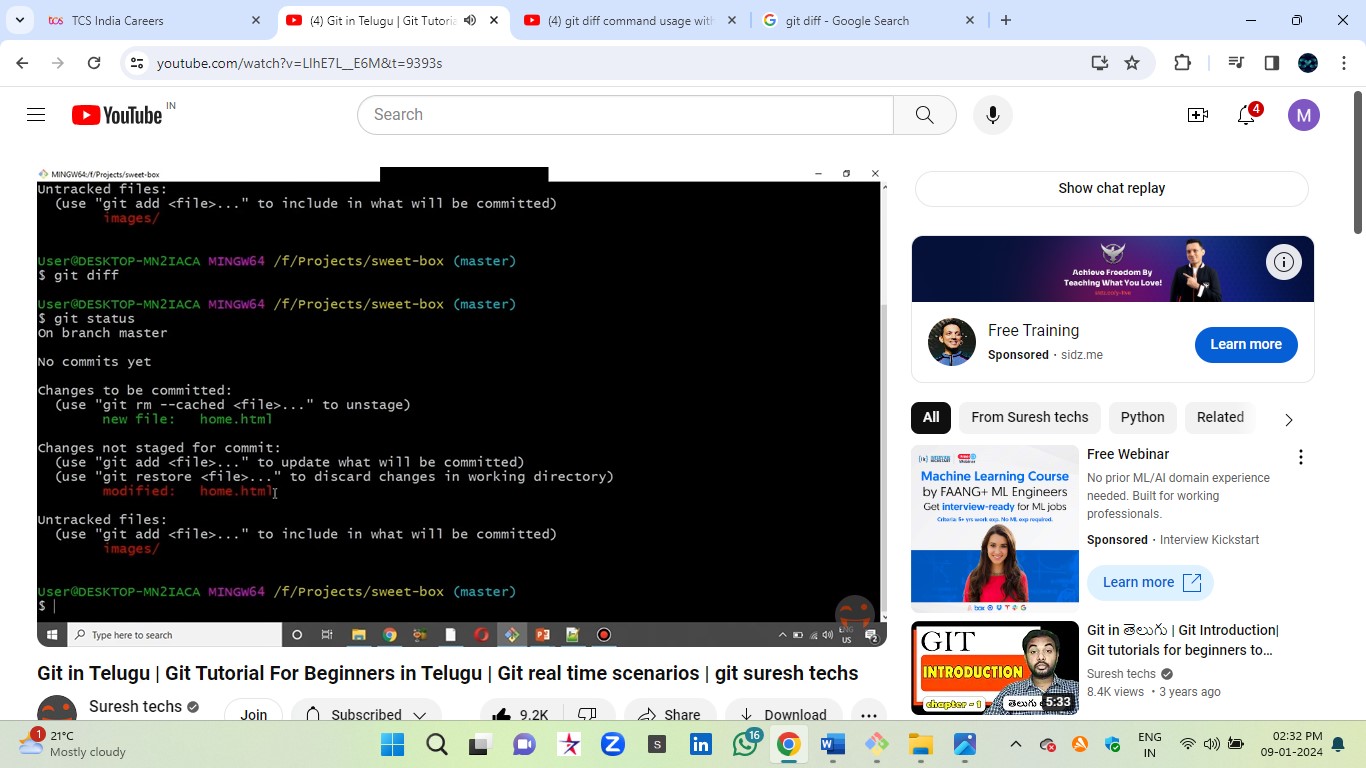
Git diff and Git status: -

* It shows the status of changes as compared to the last commit (git status)
* It shows the differences between the working directory, the staging area (index), and the last commit (git diff)

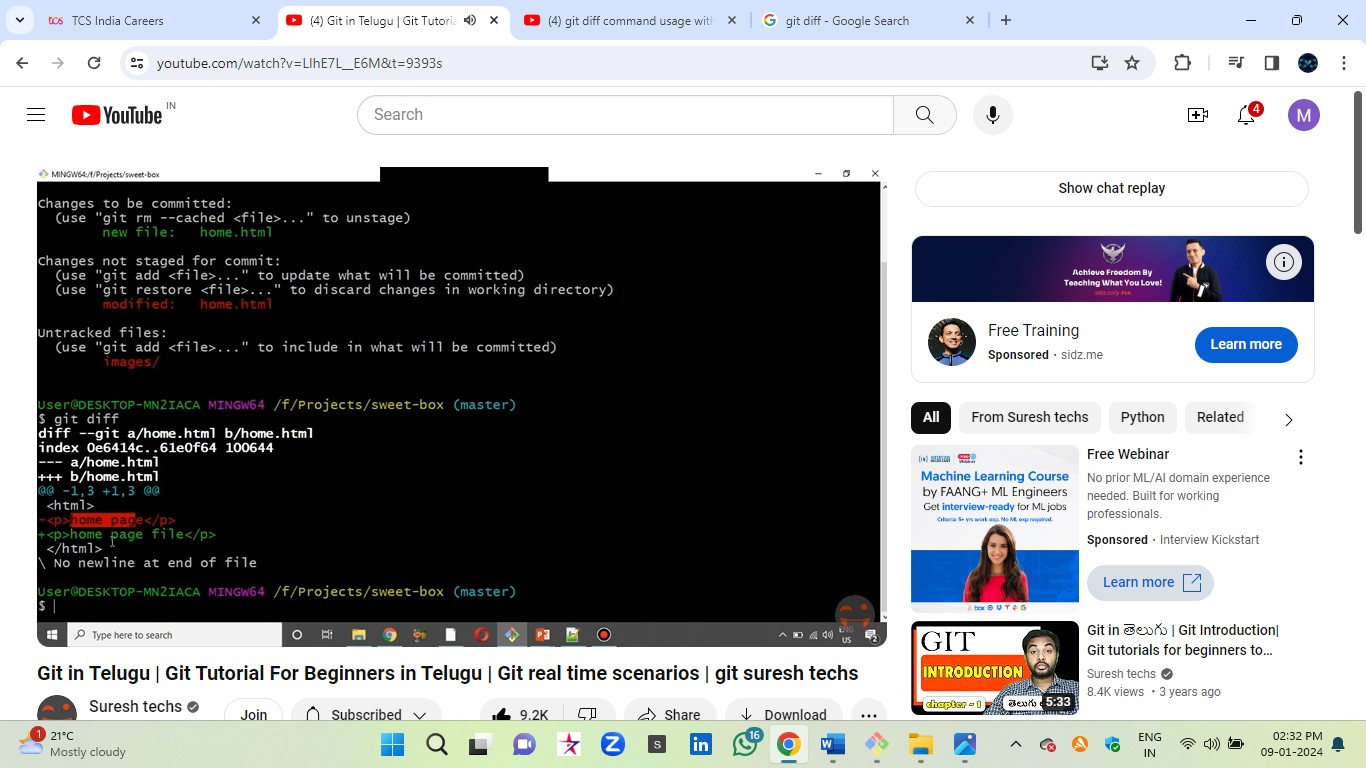
Eg:-

* Mkdir <folder name>
* Cd <folder name>
* Touch f1 and f2
* Vi <content>
* Git init (empty directory)
* Git add f1 (stage area)
* Vi f1 (modify some content)
* Git status
* Git diff (diff means it will show the changes of the content)
* Git diff --staged (which file is stage it will show the content)
* Git diff –cached
* 
* 





Git diff



**…………………………………………………GitDiff…………………………………………...**

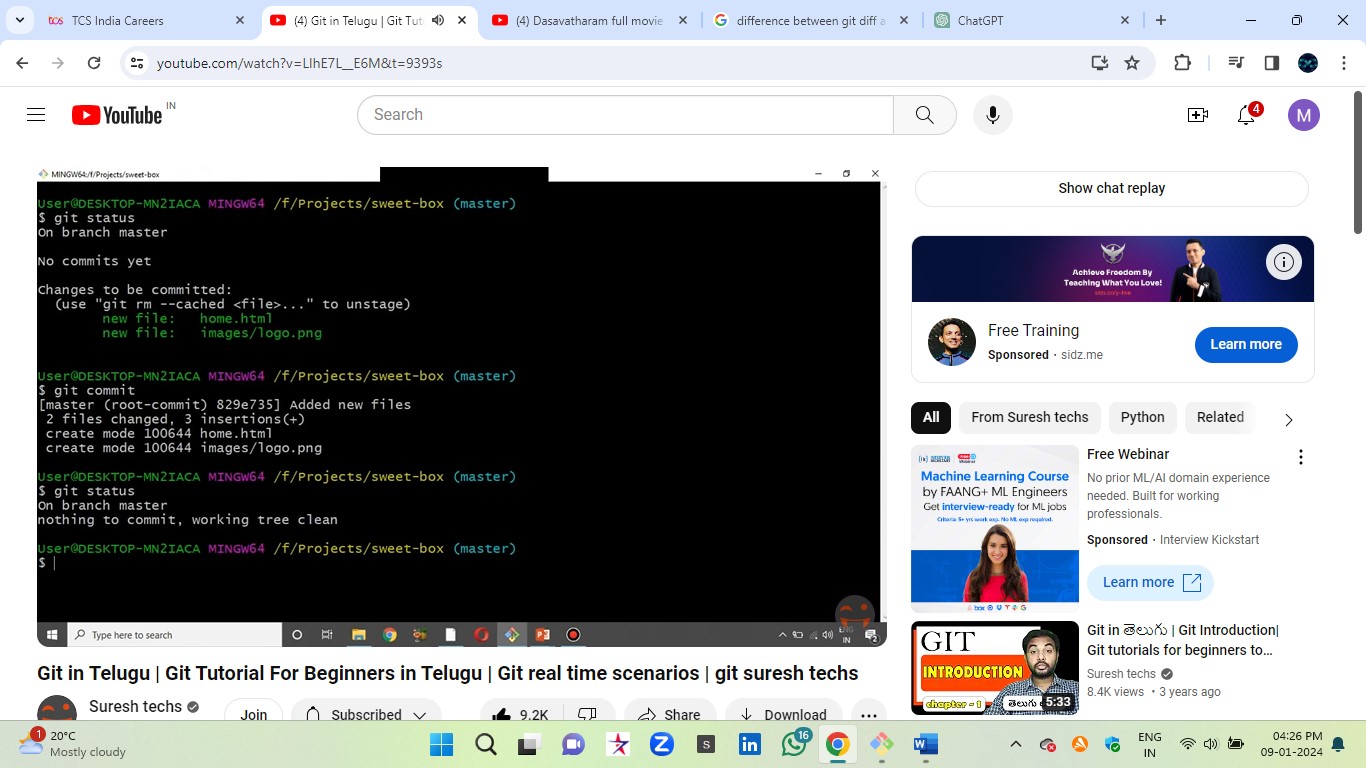
**Step9-------Git Commit and Git Log**

If you push the code into remote server before that commit message required

Def:- Commits are created with the git commit command to capture the state of a project at that point in time.

Commands: -

* Git add <filename>
* Git commit -m <message>
* git reset --hard HEAD^ (Delete commit mess)
* git commit -s
* git status
* git log (check commit history)
* git help -–all



Git Log: - (Status of the project)

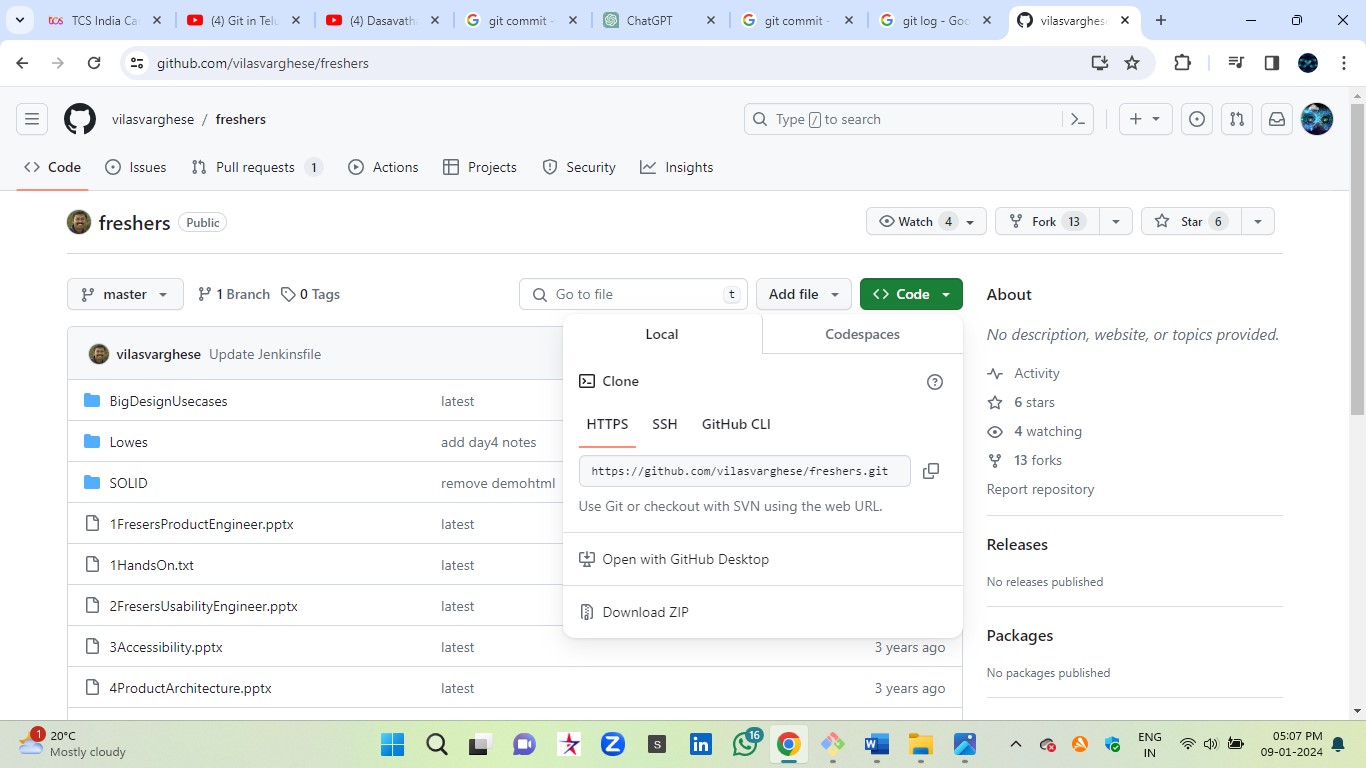
Eg:- A person worked something on project and again B person taken same project from A person that time it will use git log

Def:-  The **git log** command is used to display the commit history of a Git repository. It shows a chronological list of commits, including information about each commit such as the commit hash, author, date, and commit message.

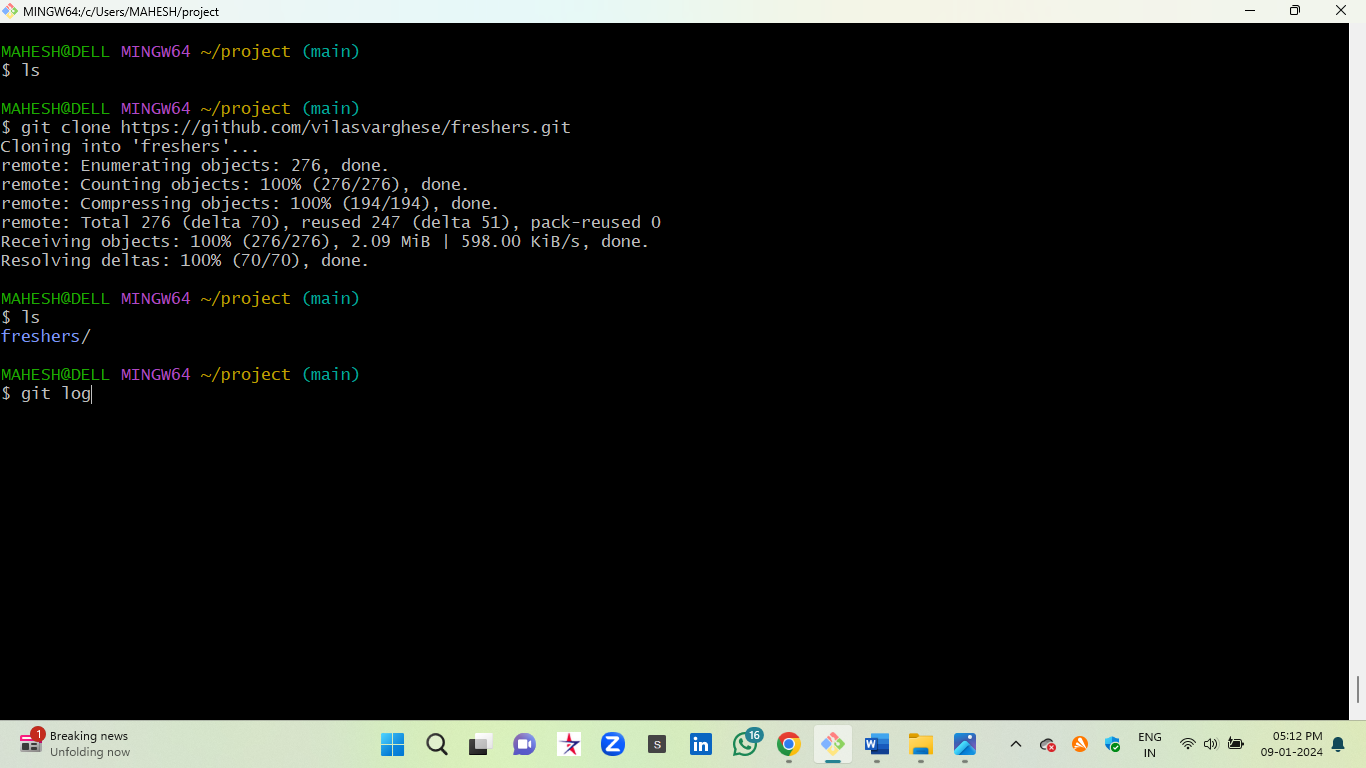
Steps: - (commands)

* Cloning code from remote to local
* Git clone <url>
* Git status
* Git log (commit history)
* Git log -p (each commit and changes)
* Git log -2 or -3 or -4 (few commits only showing)
* Git log --online (Short information)
* Git log --pretty=short
* Git log --pretty=full
* git log --graph --oneline --all

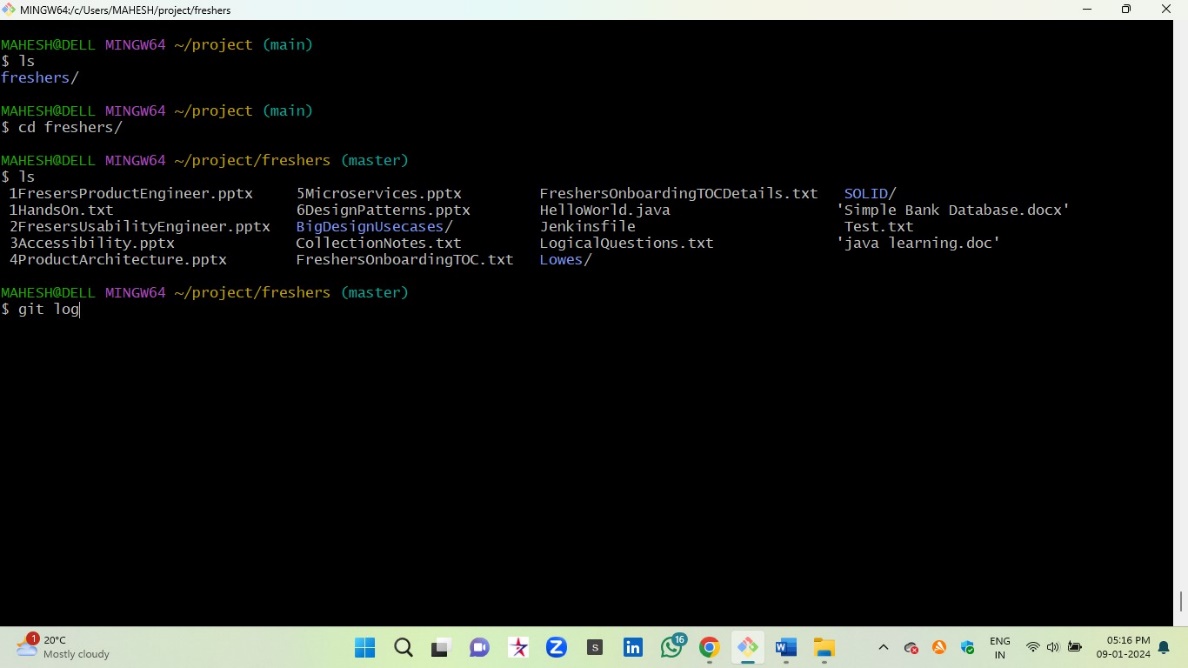
1.

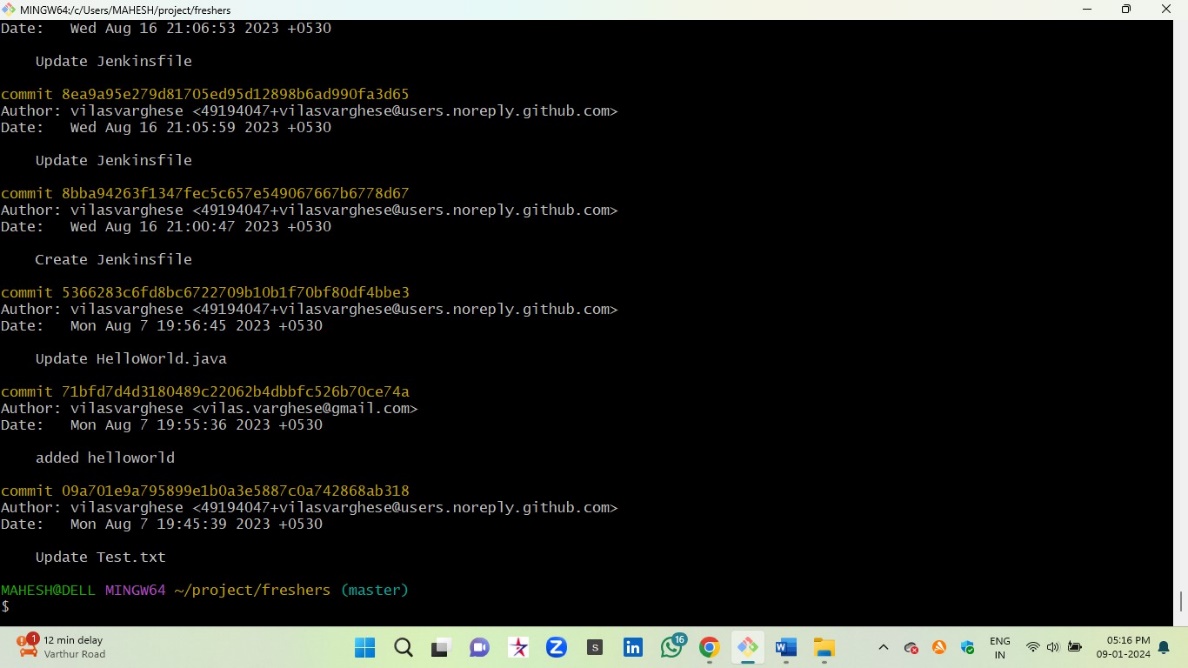


2.

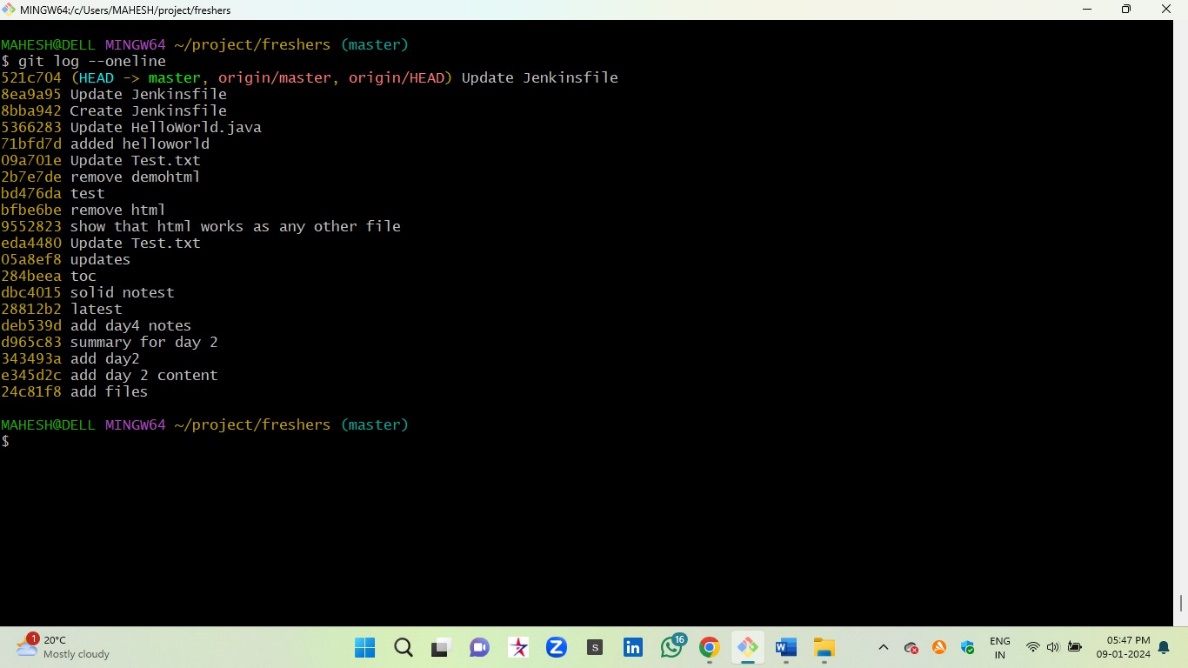


1. git log





1. git log --oneline



…………………………………….…Gitlog and git commit……………………………………

Topic 10-------Git branching

Def:-

* In Realtime the final code in master branch only
* In git default branch is master
* Rm -rf\* (file remove)
* Create different branch on local and push to remote server
* Merging branches in git hub (using pull request)
* We can create file and commit and edit in GitHub easily
* Or upload a file in GitHub directly (commit changes)

