GIT

Thursday, August 21, 2025 2:24 PM

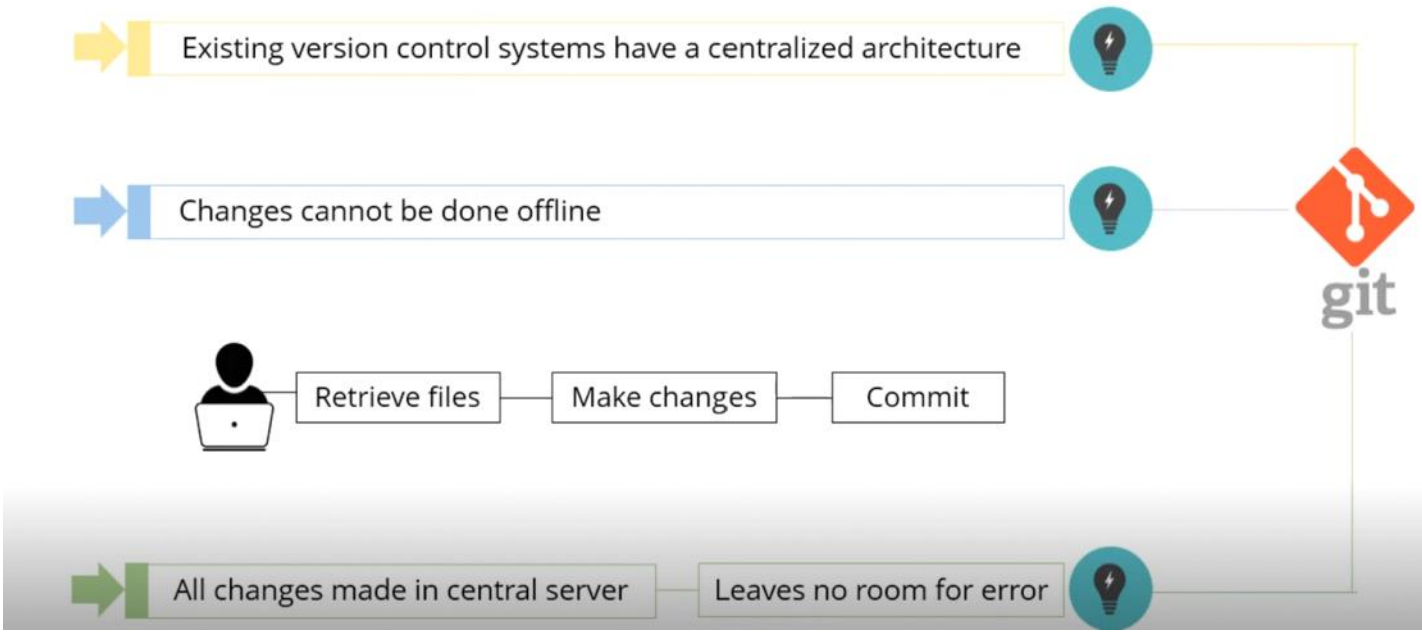
Introduction:

Requirements of Version control System:

Do have backup of your code/project.
To maintain multiple versions.

Features:
1)Branching
2)Rebasing
3)Merging
4)Cloning

Git solves these problems:


Existing version control systems have a centralized architecture
Changes cannot be done offline
git
Retrieve files
Make changes
Commit
All changes made in central server
Leaves no room for error

Installations

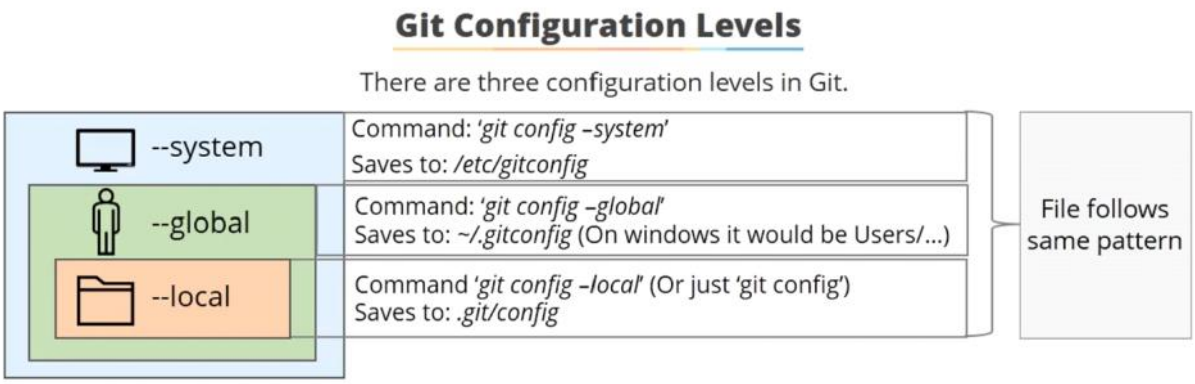
Thursday, August 21, 2025 2:42 PM

Download Git : https://git-scm.com/downloads/win

To check the version:
1)Open git bash: >> git --version

Git Configuration levels(Basics)

Thursday, August 21, 2025 2:44 PM



Note:

Local Overwrites global and global overwrites system level.

git config is the command Git uses to **set and manage configuration settings** that control Git's behavior.
Think of it like the **preferences/settings panel for Git**.

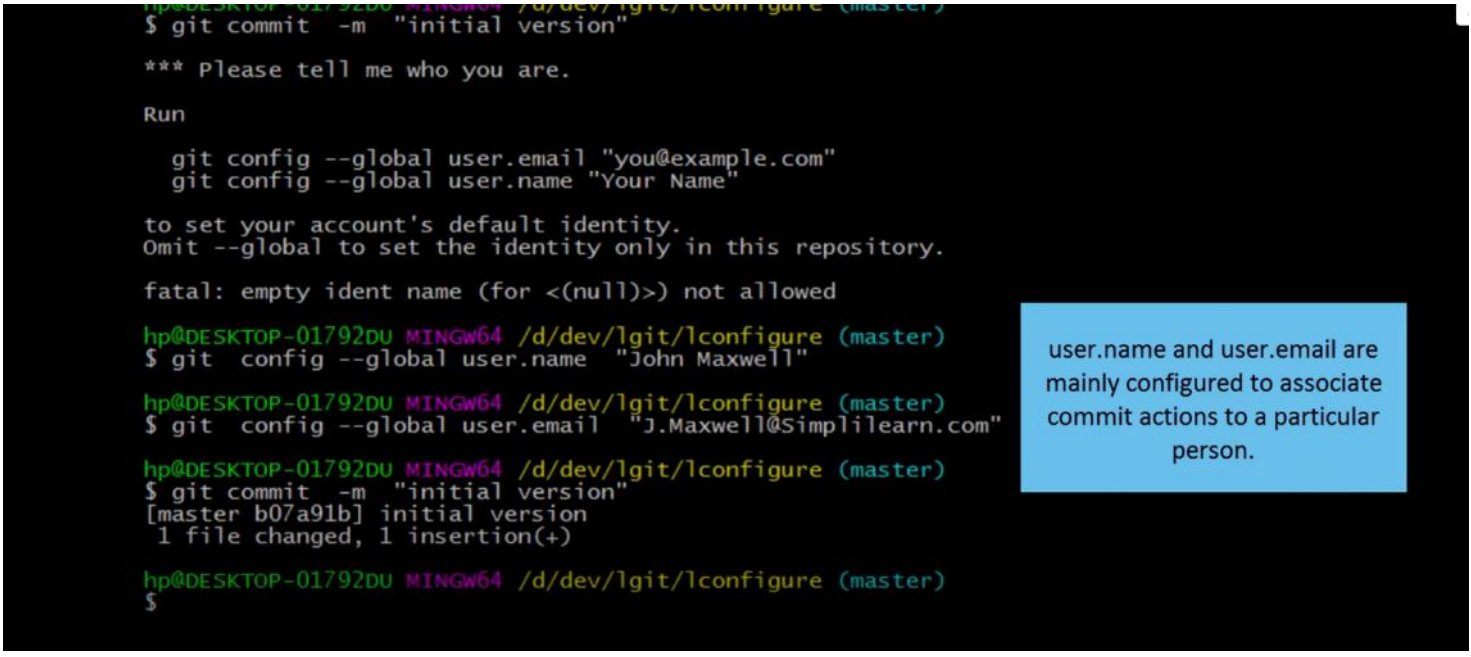
🔑 There are 3 levels of Git configuration:

1. **System level (--system)**
   - Applies to **all users** on the machine.
   - Stored in:
     - Linux/Mac → /etc/gitconfig
     - Windows → C:\ProgramData\Git\config
2. **Global level (--global)**
   - Applies to your **user account** only.
   - Stored in:
     - Linux/Mac → ~/.gitconfig
     - Windows → C:\Users\<username>\.gitconfig
3. **Local level (--local)**
   - Applies only to the **specific repository**.
   - Stored in: .git/config inside your project folder.

⚡ When Git needs a setting, it checks in this order:
Local → Global → System (local overrides global, global overrides system).

Commit Actions:

1)Git bash in any folder you want to associate commit actions.
2To commit it, you need to **add the file first**: git add filename
3)commit initial version: git commit -m "initial version"
4)set your accounts defaul identity local/global/system

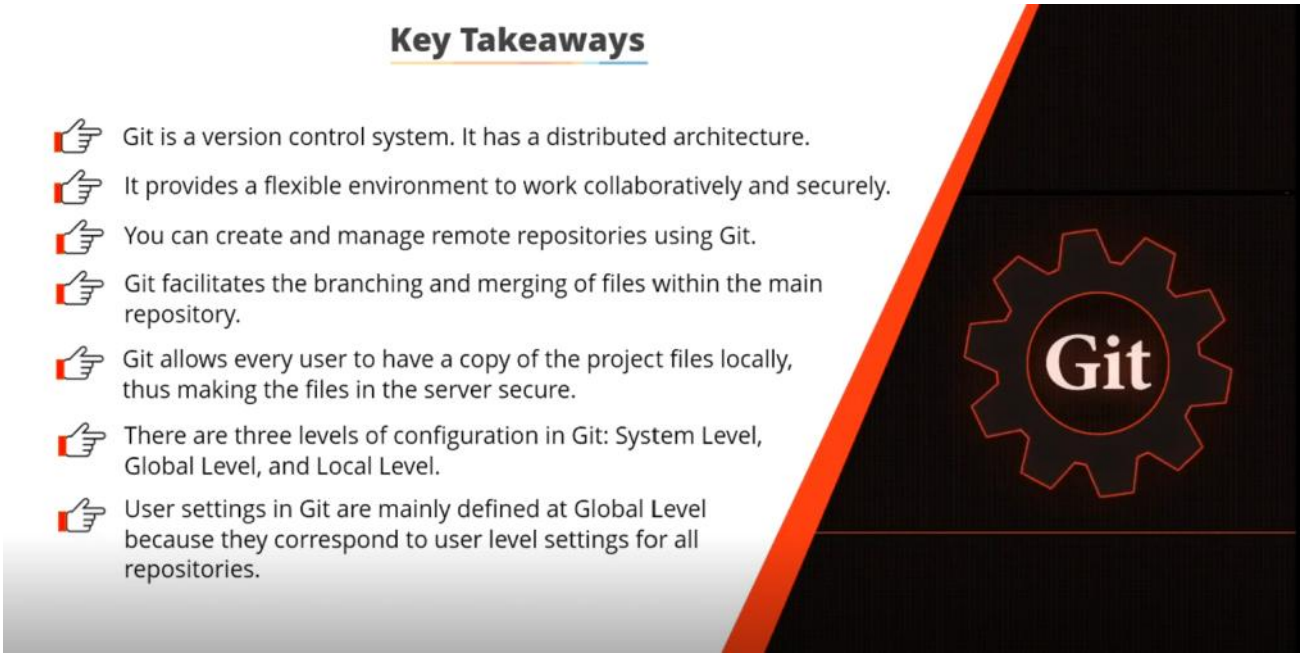


To see the configuration details :

git config --details

To do any chnages for example:/ color.ui

git config --global color.ui "changes you want to make"



Create a repository in git

Thursday, August 21, 2025 3:16 PM

Go to the folder location where you want to create a directory
$ cd "/c/Masters USA/Extra Projects"

Create a project folder
$ mkdir "git_practice"

Go inside the empty project
$ cd "git_practice"

Initialize empty git repository
$ git init

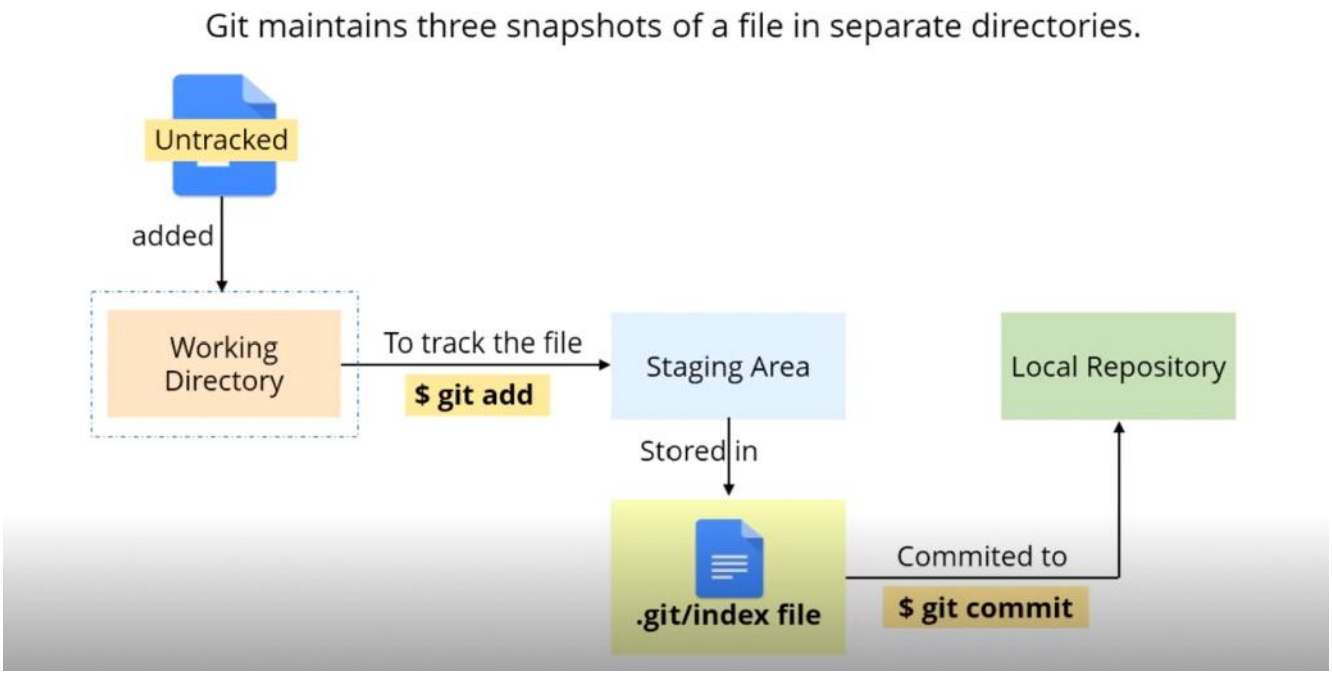
Check the contents of git directory:
ls -la

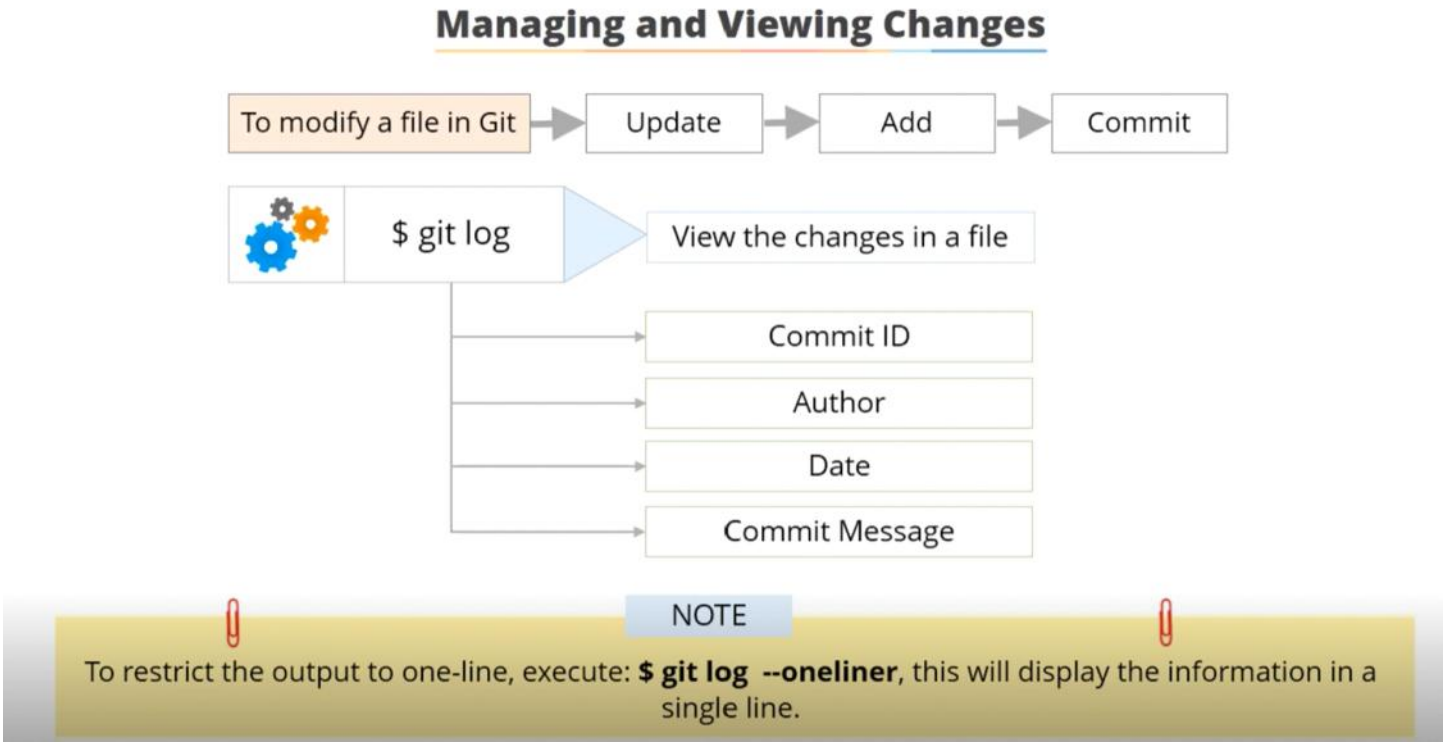
to check the status :
git status

To intialize and create a repo
git init git_practice

Git Workflow

Thursday, August 21, 2025 3:33 PM





Demo:

To check number of tracked files :

$ ls -lrt

To stage a file for git :

$ git add "file_name"

To add all file:
$ git add .

To commit the file:
$git commit

If you don’t need the msg to pop up about committing the files:
$git commit -m

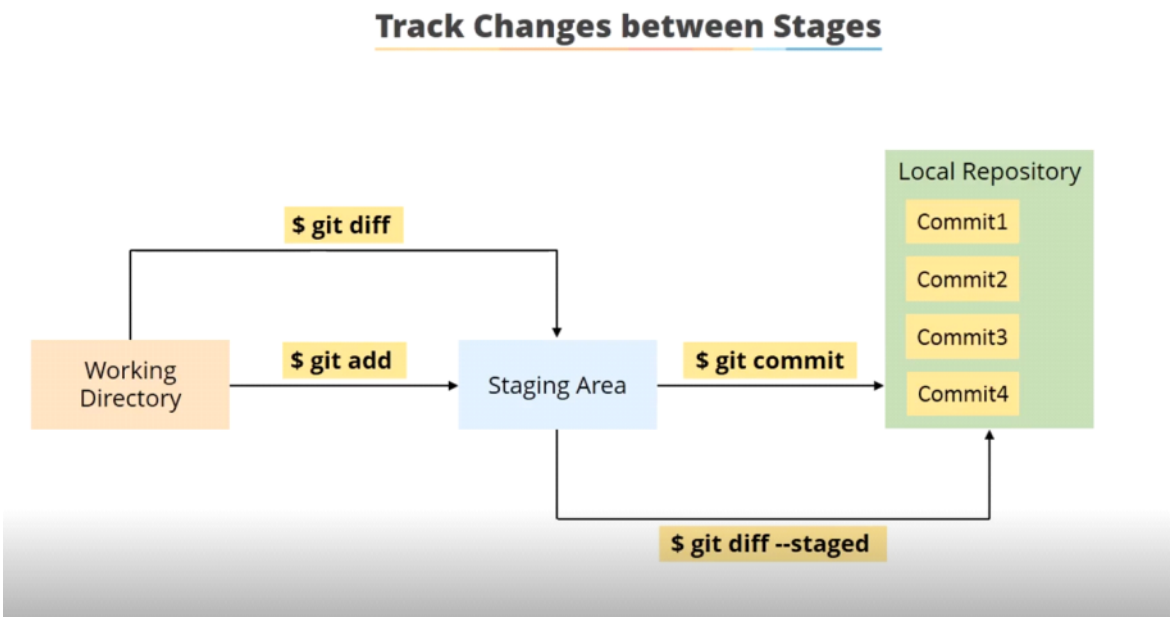
to see history of associated commits.
git log

Make sure **no Git processes are currently running** (close editors, terminals doing Git).
Then, delete the lock file manually:

rm -f "C:/Masters USA/Extra Projects/git_practice/.git/index.lock"

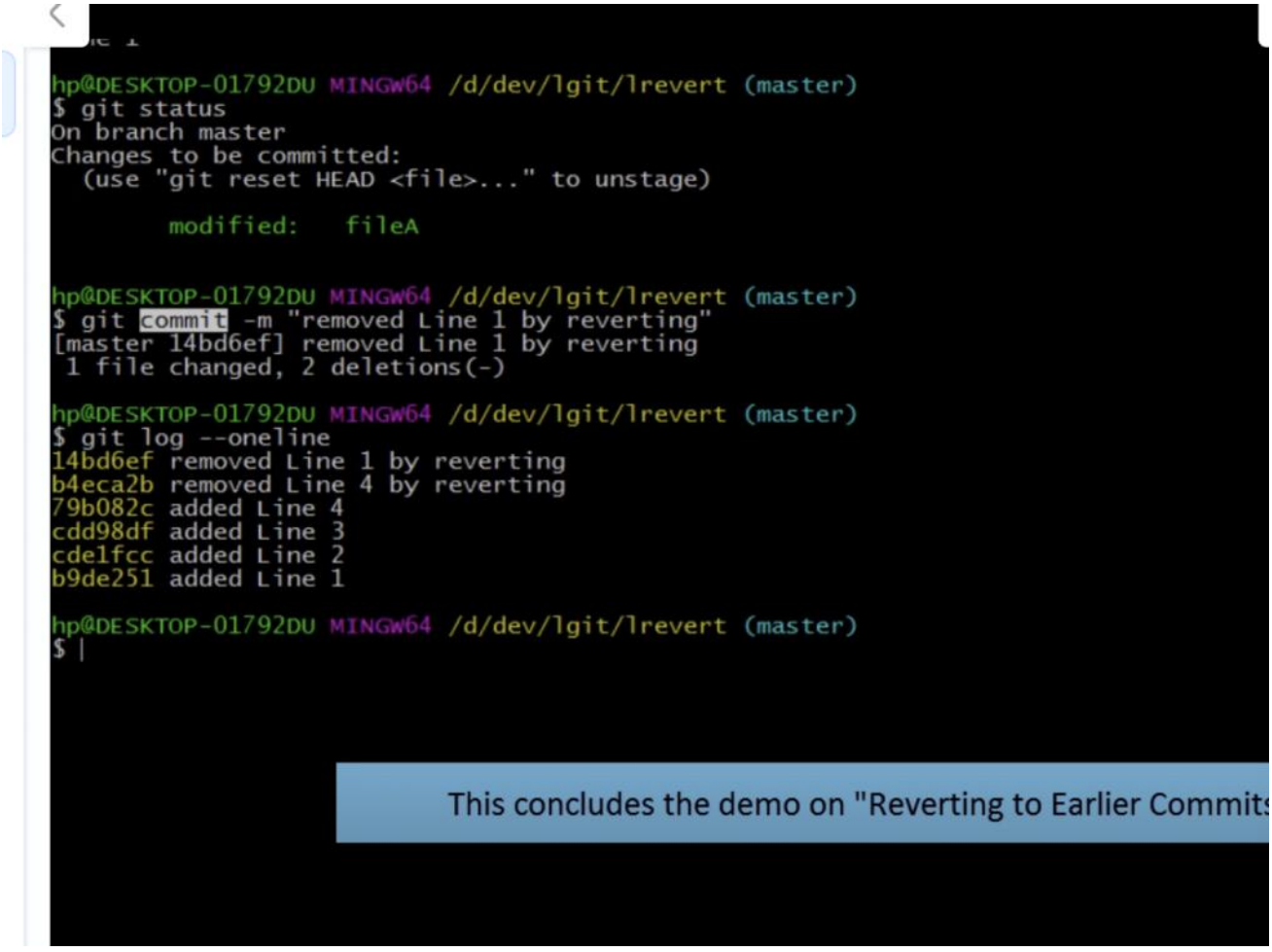
Tracking Changes

Thursday, August 21, 2025 5:25 PM



Lets Say if you have a html file
to check the status use
git status

open the file commit changes
Will show you changes.
git diff file name

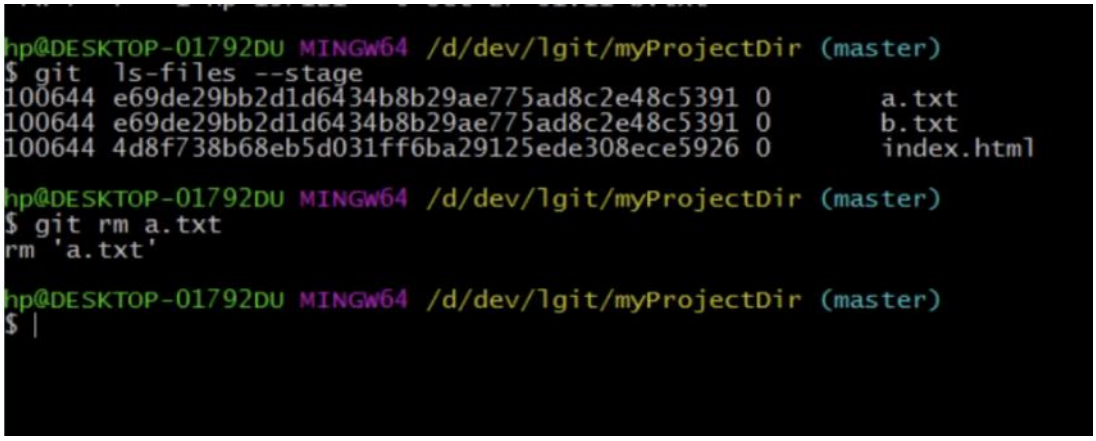


How to delete git files:

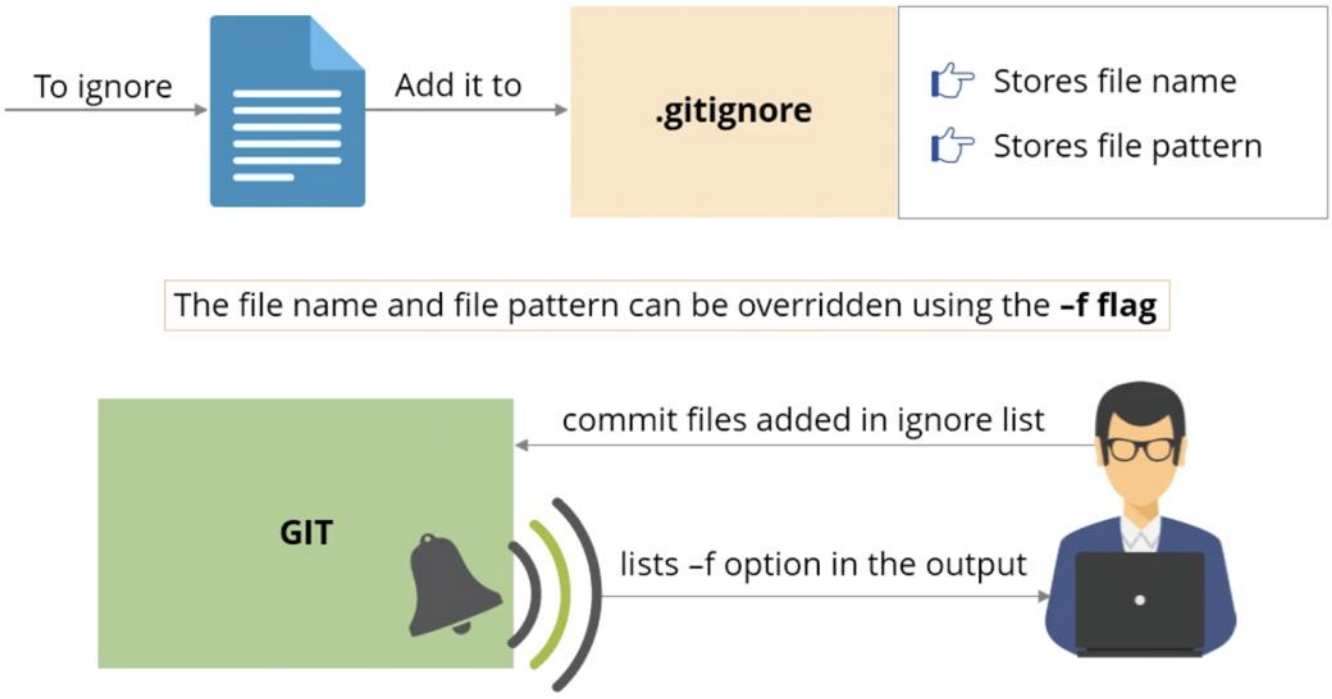
Only delete from staging area:
$git rm --cached<filename>

Delete from staging area and from repo
$git rm<filename>

Delete a particular file:



How to ignore a file:



Modify and track a file
echo "modify a.txt" >> a.txt

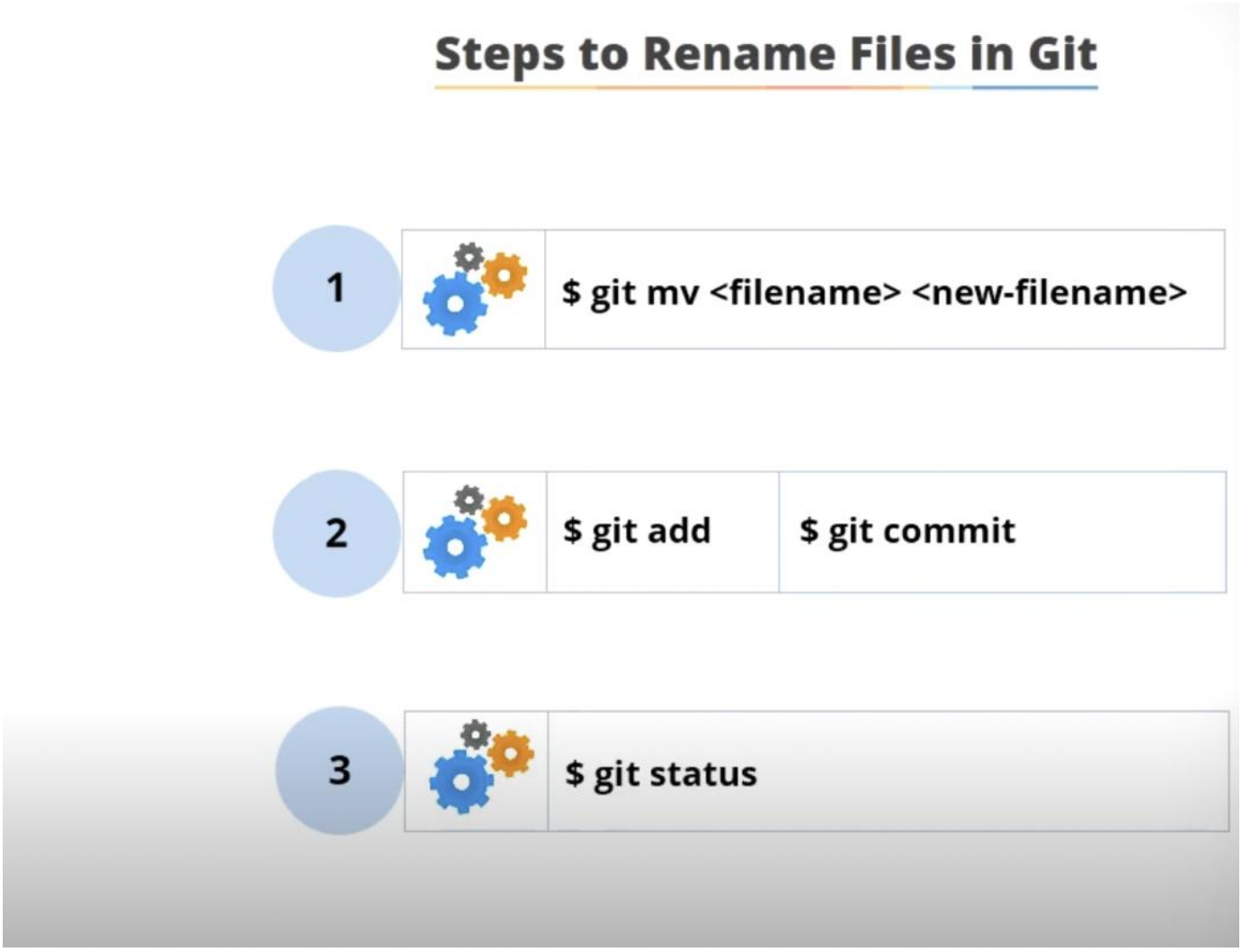
Ignore a file:
vi .gitignore

See catalog of all commits

>> git log --online

Rename Files

Saturday, August 23, 2025 5:14 PM



Change the name:
git mv a.txt aa.txt

then :
git add .

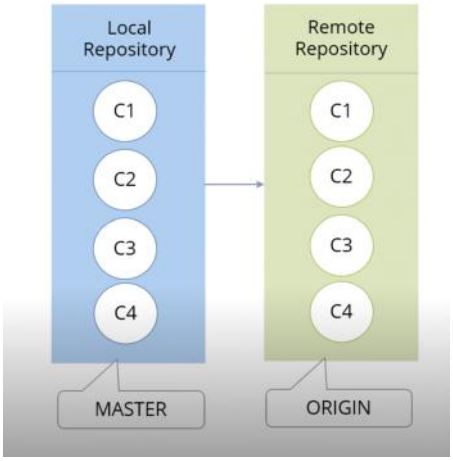
commit changes :
git commit -m "a changed to aa"

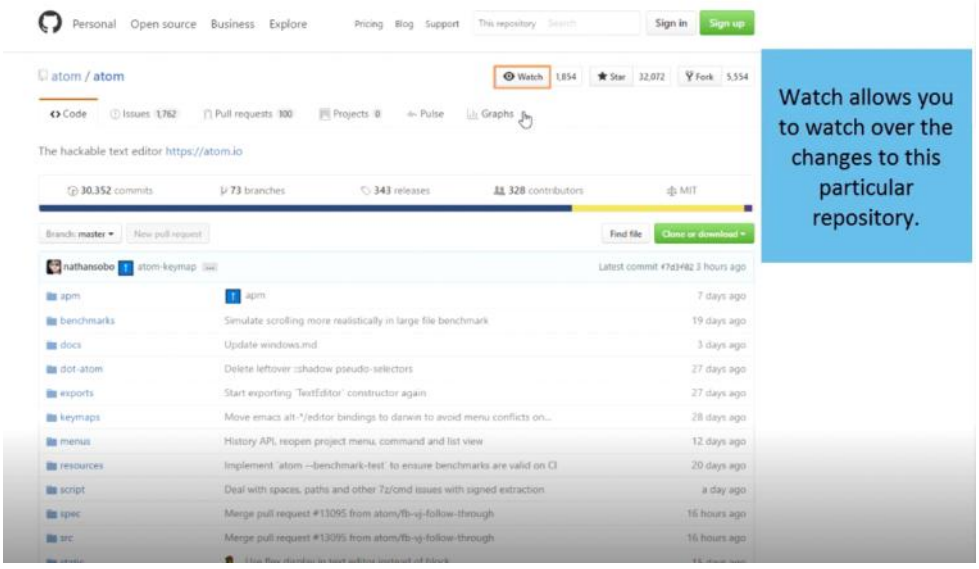
git status

GitHub

Saturday, August 23, 2025 5:53 PM

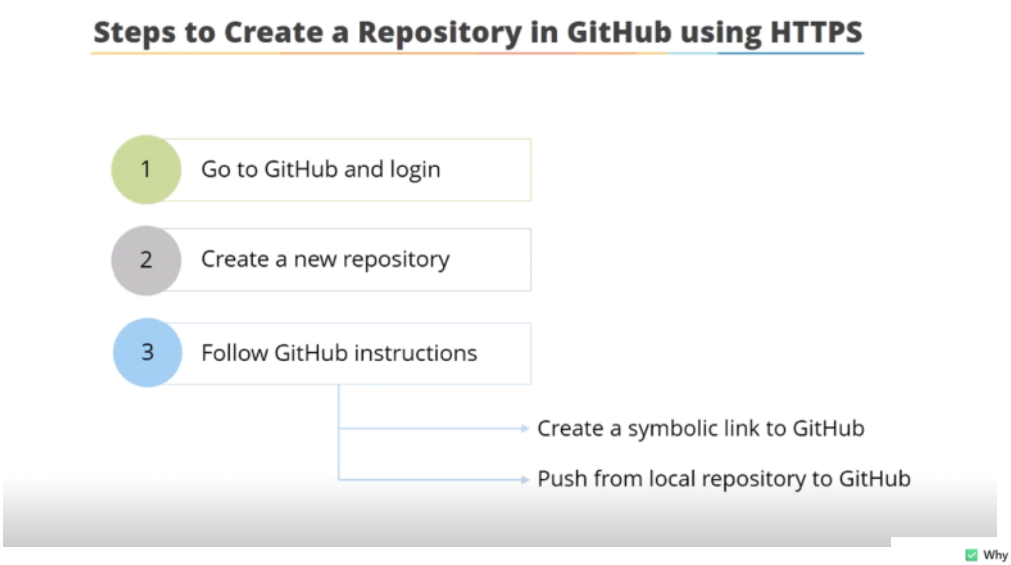
It is a web based hosting service for git repositories.





Create Repositories (HTTPS Protocol)

Sunday, August 24, 2025 7:59 AM



Initialize git:
$git init

Stage all the files:
$git add .

Commit:
$git commit -m "nothing to commit"

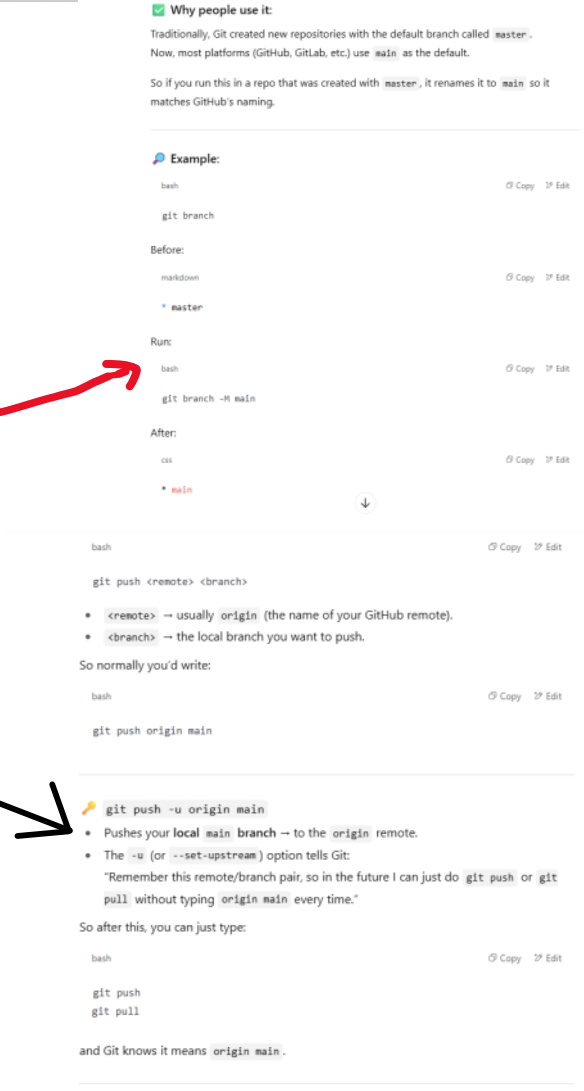
Open github and create a new repo:

Add URL of the github repo where you want to push the folder:
$git remote add origin <url>

see remotes linked to your local Git repository (optional)
$git remote -v

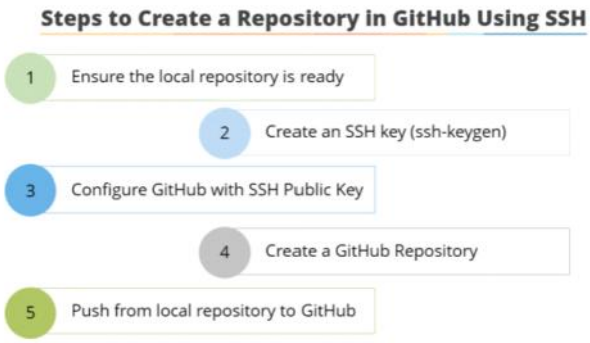
renames your current branch to main
$git branch -M main

Push repo:
$git push -u origin main



Create Repositories (SSH Protocol)

Sunday, August 24, 2025 8:28 AM

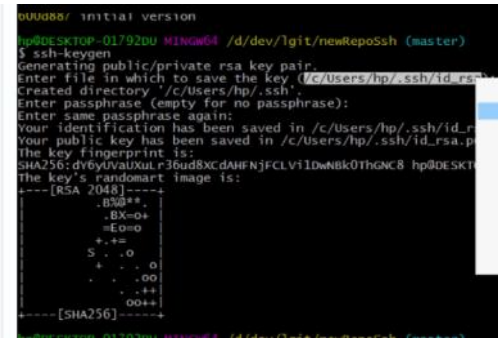


Initialize git:
$git init

Stage all the files:
$git add .

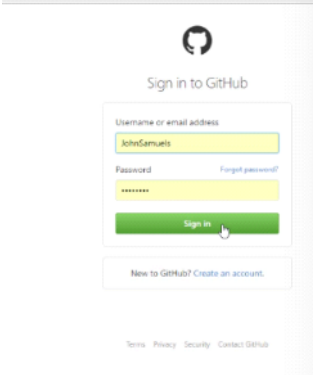
Commit:
$git commit -m "nothing to commit"
Git status

get key for ssh:
ssh -keygen

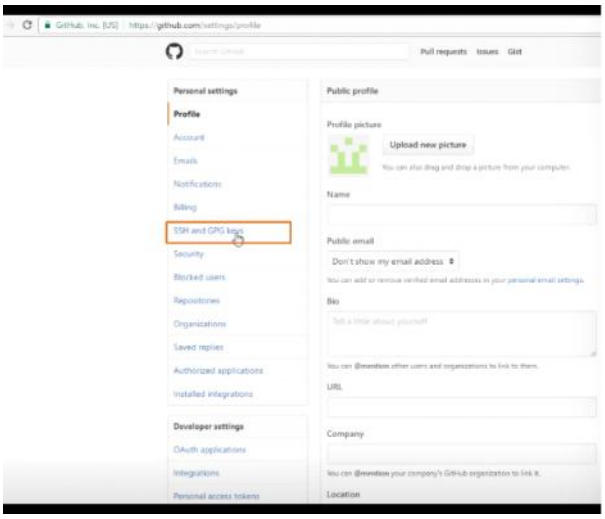


Copy the location of the key.

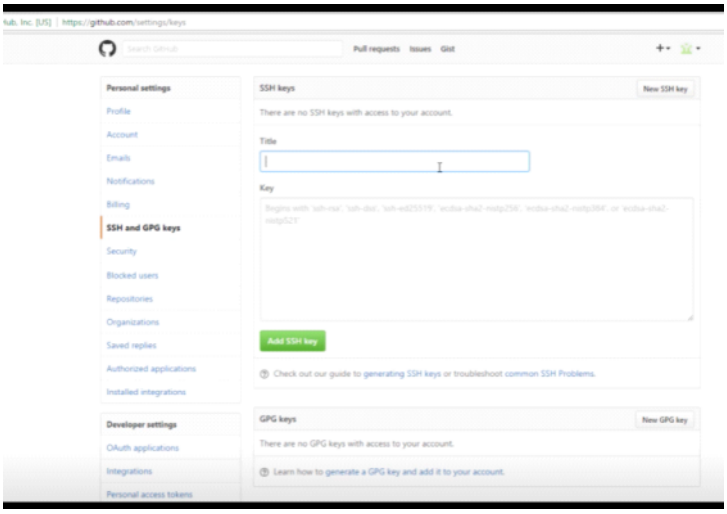
Go to github page and login:

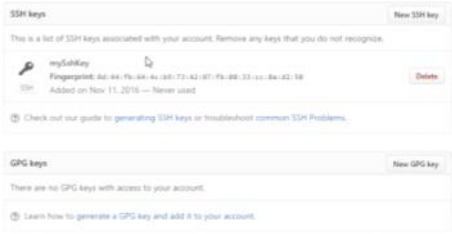


Go to view profile and more>settings>SSH and GPG Keys:

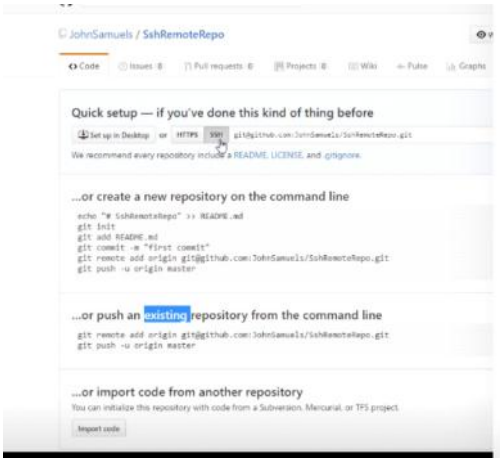


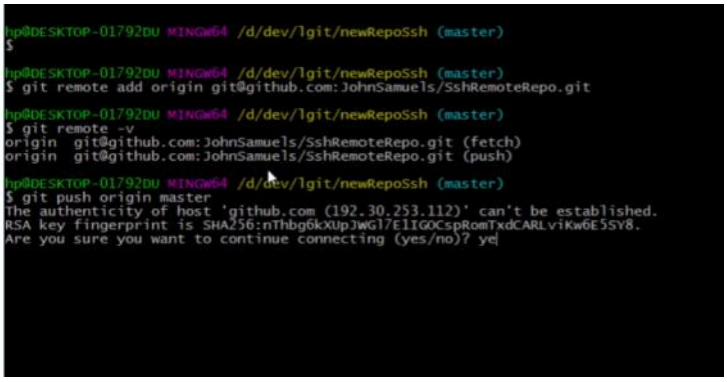
Add new SSH key to your account:





Create new repo in github:
click on SSH protocol.




hp@DESKTOP-01792DU MINGW64 /d/dev/lgit/newRepoSsh (master)
$
hp@DESKTOP-01792DU MINGW64 /d/dev/lgit/newRepoSsh (master)
$ git remote add origin git@github.com:JohnSamuels/SshRemoteRepo.git
hp@DESKTOP-01792DU MINGW64 /d/dev/lgit/newRepoSsh (master)
$ git remote -v
origin git@github.com:JohnSamuels/SshRemoteRepo.git (fetch)
origin git@github.com:JohnSamuels/SshRemoteRepo.git (push)
hp@DESKTOP-01792DU MINGW64 /d/dev/lgit/newRepoSsh (master)
$ git push origin master
The authenticity of host 'github.com (192.30.253.112)' can't be established.
RSA key fingerprint is SHA256:nThbg6kXUpJWG17E1IGOCspRomTxdCARLviKw6E5SY8.
Are you sure you want to continue connecting (yes/no)? ye

Git and GitHub collaboration

Sunday, August 24, 2025 8:44 AM

Pull commits from remote repo:

Open GitHub repo:

Add a file/ edit a file in the repo:

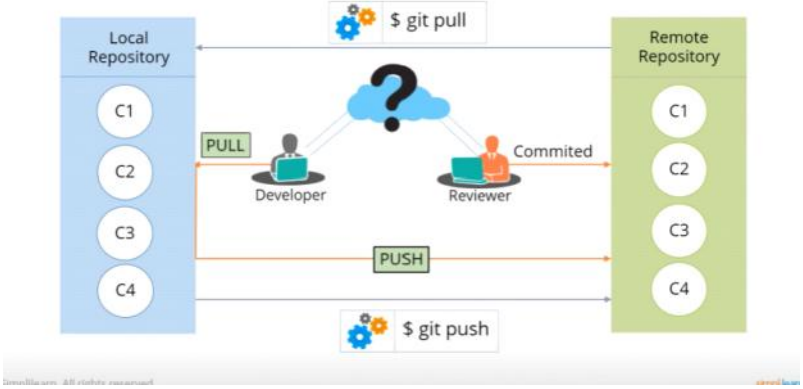
open git locally and open gitbash:
$git pull origin main

```
ommor@Ommore5807 MINGW64 /c/Masters USA/Extra Projects/git_practice (main)
$ git pull origin master
fatal: couldn't find remote ref master

ommor@Ommore5807 MINGW64 /c/Masters USA/Extra Projects/git_practice (main)
$ git pull origin main
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 957 bytes | 79.00 KiB/s, done.
From https://github.com/Ommore5807/Git_course
 * branch            main       -> FETCH_HEAD
   6498b54..e1b3e45  main       -> origin/main
Updating 6498b54..e1b3e45
Fast-forward
 README.md | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 README.md

ommor@Ommore5807 MINGW64 /c/Masters USA/Extra Projects/git_practice (main)
$
```

Remote and Local Collaboration:

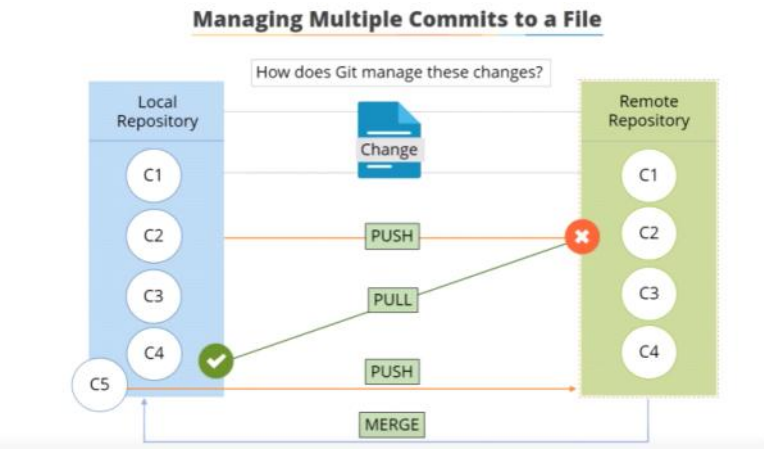


1)Add a file locally or remotely

if locally: commit and push
$ git push -u origin main
if globally: pull
$git pull origin main

Managing Multiple Commits to a file:

If there is a commit in global and commit locally. But commit globally have occurred before, then one need to pull that first before pushing the commits to have entire code.



Demo:

I added a pdf file to Local repo:

Git_Course_Notes.pdf

I made some changes in ReadMe.md file in remote repo:

How to sync both changes:

Try pushing the changes from local to remote:

```
ommor@Ommore5807 MINGW64 /c/Masters USA/Extra Projects/git_practice (main|MERGI
G)
$ git push origin main
To https://github.com/Ommore5807/Git_course.git
 ! [rejected]        main -> main (non-fast-forward)
error: failed to push some refs to 'https://github.com/Ommore5807/Git_course.g
'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. If you want to integrate the remote changes,
hint: use 'git pull' before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

ommor@Ommore5807 MINGW64 /c/Masters USA/Extra Projects/git_practice (main|MERGI
G)
$ git pull -m origin main
error: unknown switch `m'
usage: git pull [<options>] [<repository> [<refspec>...]]

    -v, --[no-]verbose    be more verbose
    -q, --[no-]quiet      be more quiet
```

It will send a rejected message because one need to sync the remote commits first.

Pull the remote commits:
$ git pull origin main

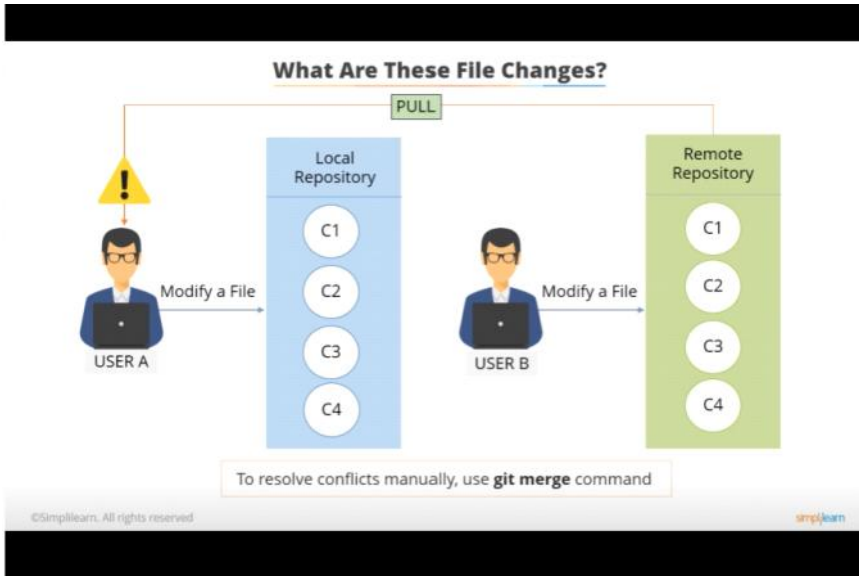
Stage the files in local repo:
$git add "file name" --> e.g. $git add "Git_Course_Notes.pdf"

Commit the file changes:
$git commit -m "<comment> --> e.g. $git commit -m "Added notes"

Commit changes for pulled file:
$git commit -m "<comment> --> $git commit -m "file downloaded"

Push the changes:
$git push origin main

Changing the same line of same file Remote and Locally:

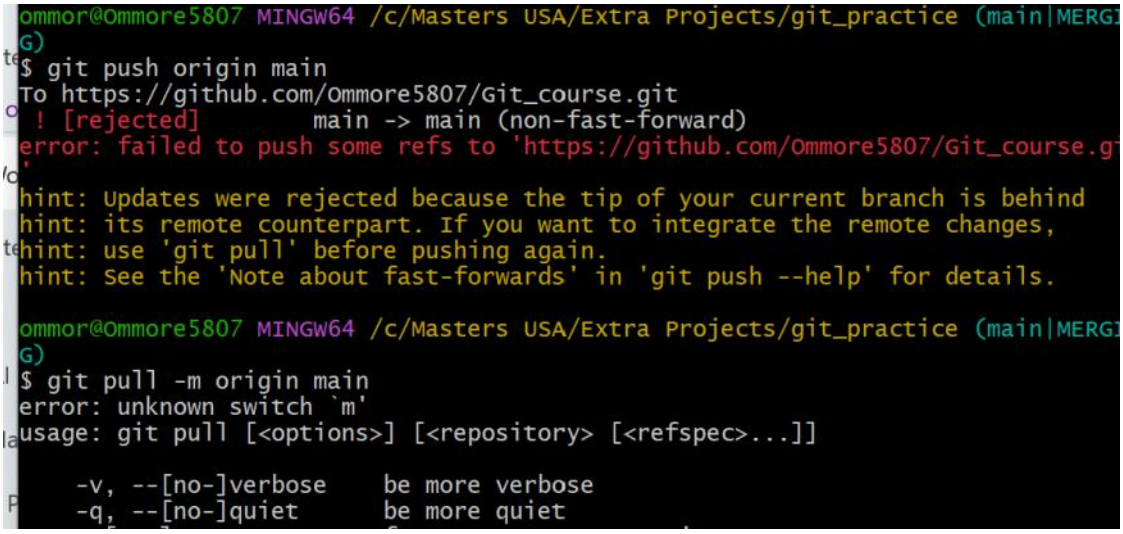


Demo:

I edited a readme file :
and
I made some changes in ReadMe.md file in remote repo:

How to sync both changes:

Try pushing the changes from local to remote:



It will send a rejected message because one need to sync the remote commits first.

Pull the remote commits:
$ git pull origin main

Stage the files in local repo:
$git add "file name"

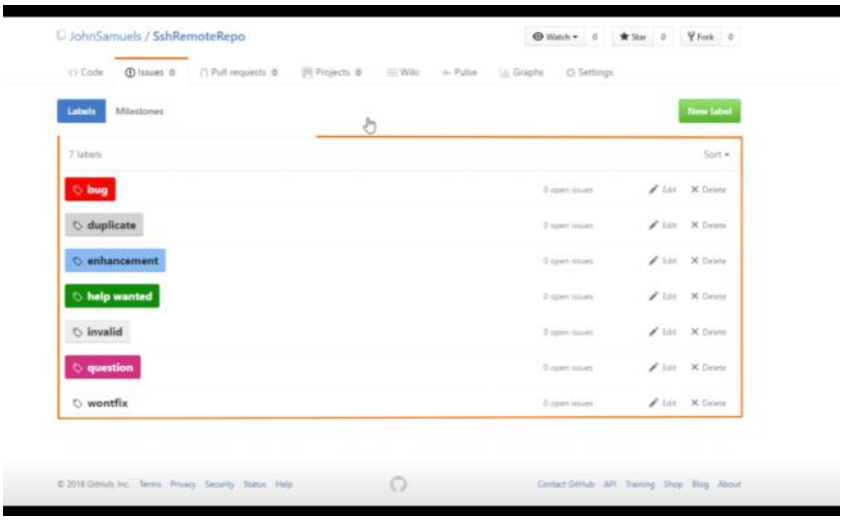
Commit the file changes:
$git commit -m "<comment> --> e.g. $git commit -m "changes made globally"

Commit changes for pulled file:
$git commit -m "<comment> --> $git commit -m "local changes are finalized"

Push the changes:
$git push origin main

Creating Tracking Issues

Sunday, August 24, 2025 9:32 AM



◇ 1. What is a GitHub Issue?

- An **issue** is GitHub's way to track tasks, bugs, feature requests, or discussions.
- Each issue has:
  - **Title & description** (explain the problem/task)
  - **Labels** (categorize issues, e.g., bug, enhancement)
  - **Assignees** (who is responsible)
  - **Milestones** (group issues into a release/project goal)
  - **Comments** (discussion space).

◇ 2. Creating & Managing Issues

- Create an issue from the **"Issues" tab** in a repo.
- Use **Markdown** in descriptions for formatting, checklists, code snippets.
- Use **labels** to prioritize:
  - bug, help wanted, good first issue, documentation, etc.
- Assign issues to collaborators so everyone knows ownership.

◇ 3. Linking Issues with Commits & Pull Requests

- Mention an issue number in a commit or PR using #issue_number:

  git commit -m "Fix login bug (#12)"
- In a PR/commit message, use keywords to automatically close issues when merged:
  - Fixes #12
  - Closes #34
  - Resolves #56
- This helps keep issues in sync with code changes.

◇ 4. Tracking Progress

- **Milestones** → group related issues for a specific release/goal.

- **Projects/Boards** → visualize issues in Kanban style (To Do → In Progress → Done).
- **Issue States**:
    - Open → pending work.
    - Closed → resolved, won't be worked on, or duplicate.

◇ 5. Notifications & Collaboration

- Watch a repo to get notifications about new issues/updates.
- Mention teammates with @username in comments.
- Use GitHub Actions to automate issue tracking (e.g., auto-assign labels).