

## Introduction:

### What is Git?

Git is a version control system.



Speed



Data integrity



Security

### 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



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:

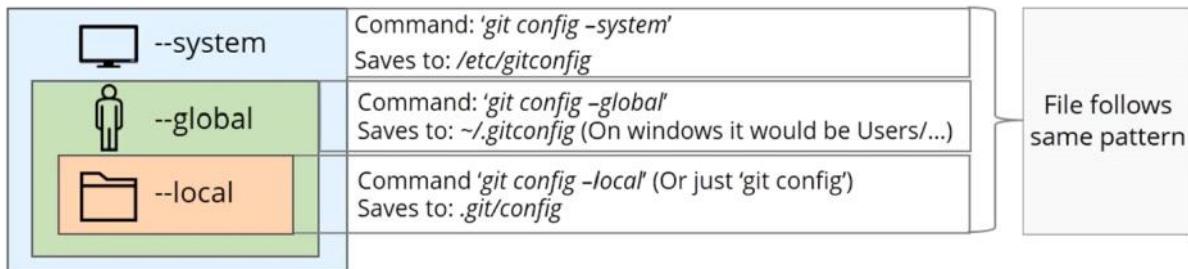
1) Open git bash: >> git --version

# Git Configuration levels(Basics)

Thursday, August 21, 2025 2:44 PM

## Git Configuration Levels

There are three configuration levels in Git.



### 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.

2) To 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 default identity local/global/system

```

hp@DESKTOP-01792DU MINGW64 /d/dev/lgit/lconfigure (master)
$ git commit -m "initial version"
*** Please tell me who you are.

Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.

fatal: empty ident name (for <(null)>) not allowed

hp@DESKTOP-01792DU MINGW64 /d/dev/lgit/lconfigure (master)
$ git config --global user.name "John Maxwell"

hp@DESKTOP-01792DU MINGW64 /d/dev/lgit/lconfigure (master)
$ git config --global user.email "J.Maxwell@Simplilearn.com"
hp@DESKTOP-01792DU MINGW64 /d/dev/lgit/lconfigure (master)
$ git commit -m "initial version"
[master b07a91b] initial version
 1 file changed, 1 insertion(+)

hp@DESKTOP-01792DU MINGW64 /d/dev/lgit/lconfigure (master)
$
```

user.name and user.email are mainly configured to associate commit actions to a particular person.

### To see the configuration details :

git config --details

### To do any changes for example:/ color.ui

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

## Key Takeaways

- 👉 Git is a version control system. It has a distributed architecture.
- 👉 It provides a flexible environment to work collaboratively and securely.
- 👉 You can create and manage remote repositories using Git.
- 👉 Git facilitates the branching and merging of files within the main repository.
- 👉 Git allows every user to have a copy of the project files locally, thus making the files in the server secure.
- 👉 There are three levels of configuration in Git: System Level, Global Level, and Local Level.
- 👉 User settings in Git are mainly defined at Global Level because they correspond to user level settings for all repositories.



# 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
```

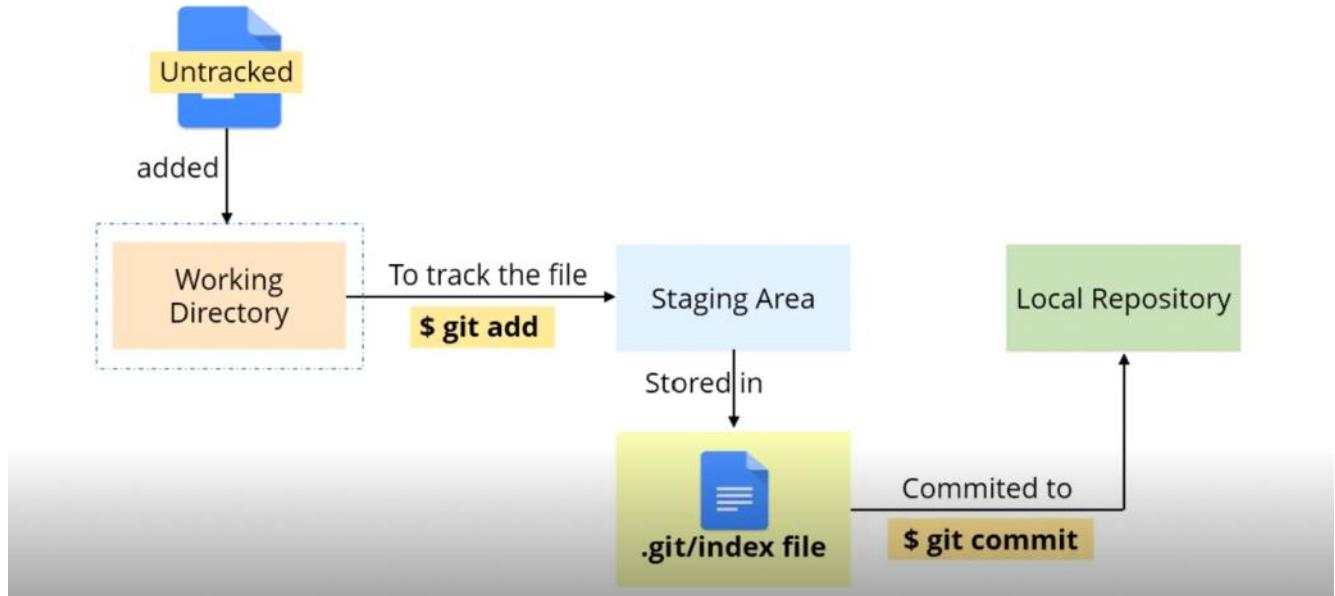
**To intialize and create a repo**

```
git init git_practice
```

# Git Workflow

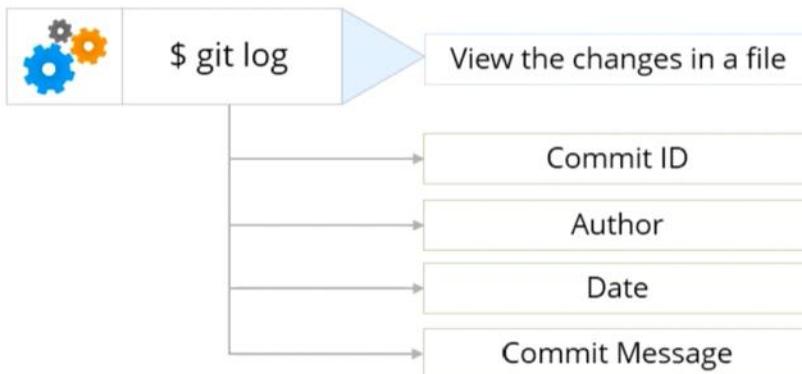
Thursday, August 21, 2025 3:33 PM

Git maintains three snapshots of a file in separate directories.



## Managing and Viewing Changes

To modify a file in Git → Update → Add → Commit



### NOTE

To restrict the output to one-line, execute: `$ git log --oneliner`, this will display the information in a single line.

## 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).

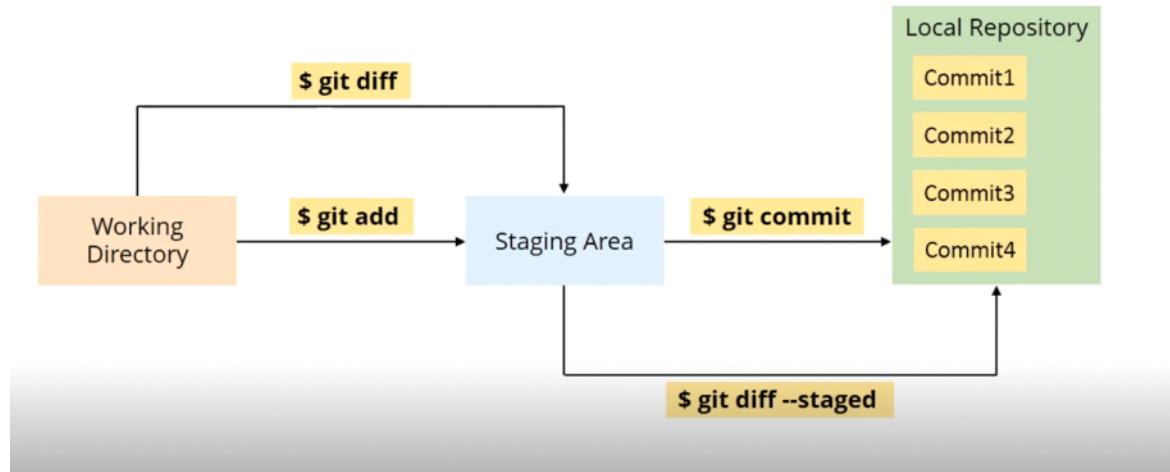
Then, delete the lock file manually:

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

# Tracking Changes

Thursday, August 21, 2025 5:25 PM

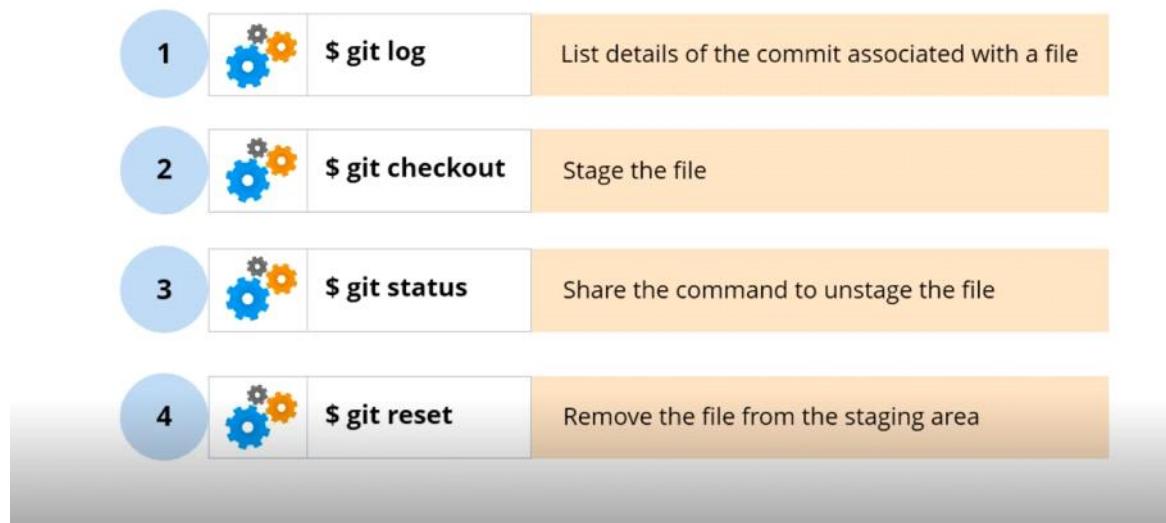
## Track Changes between Stages



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

## Steps to Revert to Earlier Commits in Git



```
hp@DESKTOP-01792DU MINGW64 /d/dev/lgit/lrevert (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   fileA

hp@DESKTOP-01792DU MINGW64 /d/dev/lgit/lrevert (master)
$ git commit -m "removed Line 1 by reverting"
[master 14bd6ef] removed Line 1 by reverting
 1 file changed, 2 deletions(-)

hp@DESKTOP-01792DU MINGW64 /d/dev/lgit/lrevert (master)
$ git log --oneline
14bd6ef removed Line 1 by reverting
b4eca2b removed Line 4 by reverting
79b082c added Line 4
cdd98df added Line 3
cdelfcc added Line 2
b9de251 added Line 1

hp@DESKTOP-01792DU MINGW64 /d/dev/lgit/lrevert (master)
$ |
```

This concludes the demo on "Reverting to Earlier Commits"

### 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:

```
hp@DESKTOP-01792DU MINGW64 /d/dev/lgit/myProjectDir (master)
$ git ls-files --stage
100644 e69de29bb2d1d6434b8b29ae775ad8c2e48c5391 0      a.txt
100644 e69de29bb2d1d6434b8b29ae775ad8c2e48c5391 0      b.txt
100644 4d8f738b68eb5d031ff6ba29125ede308ece5926 0      index.html

hp@DESKTOP-01792DU MINGW64 /d/dev/lgit/myProjectDir (master)
$ git rm a.txt
rm 'a.txt'

hp@DESKTOP-01792DU MINGW64 /d/dev/lgit/myProjectDir (master)
$ |
```

### How to ignore a file:

## How to Ignore Files in Git



The file name and file pattern can be overridden using the **-f flag**



### 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

## Steps to Rename Files in Git

1



\$ git mv <filename> <new-filename>

2



\$ git add

\$ git commit

3



\$ git status

**Change the name:**

git mv a.txt aa.txt

**then :**

git add .

**commit changes :**

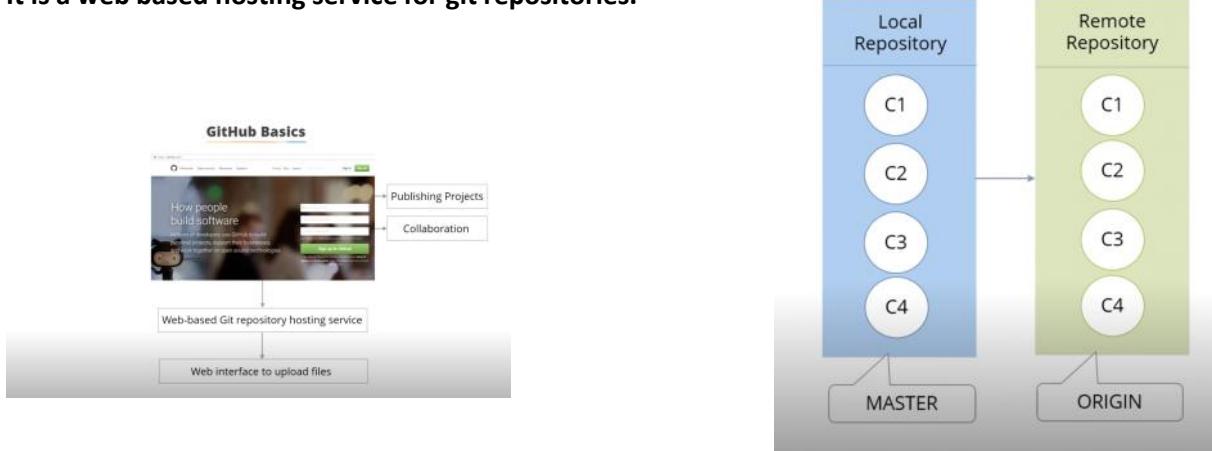
git commit -m "a changed to aa"

git status

# GitHub

Saturday, August 23, 2025 5:53 PM

**It is a web based hosting service for git repositories.**



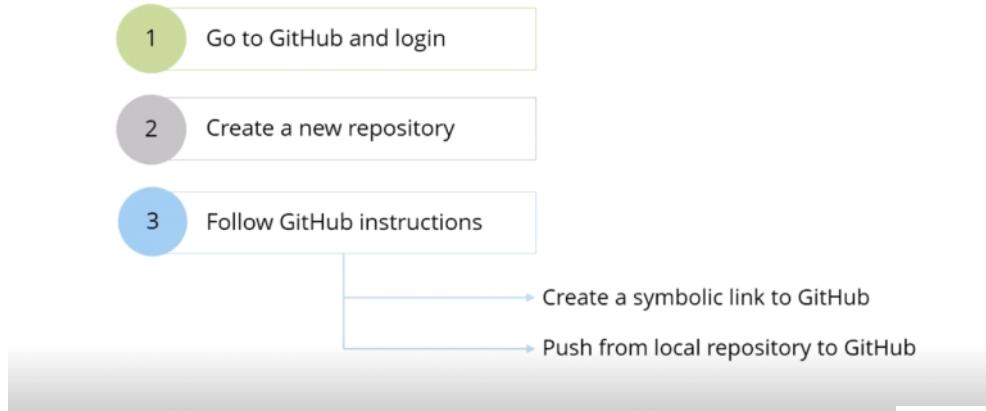
The screenshot shows the GitHub repository page for "atom / atom". The top navigation bar includes links for Personal, Open source, Business, Explore, Pricing, Blog, Support, and Sign in. The repository stats are: 30,352 commits, 73 branches, 343 releases, 328 contributors, and MIT license. A "Watch" button is highlighted. The main content area displays a list of recent commits from user "nathansobo" and "stom-keymap". A blue callout box on the right states: "Watch allows you to watch over the changes to this particular repository."

Commit	Author	Date
apm	nathansobo	7 days ago
benchmarks		19 days ago
docs		3 days ago
dot-atom		27 days ago
exports		27 days ago
keymaps		28 days ago
menus		12 days ago
resources		20 days ago
script		a day ago
spec		16 hours ago
src		16 hours ago

# Create Repositories (HTTPS Protocol)

Sunday, August 24, 2025 7:59 AM

## Steps to Create a Repository in GitHub using HTTPS



### 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
```

### Why people use it:

Traditionally, Git created new repositories with the default branch called `master`. Now, most platforms (GitHub, GitLab, etc.) use `main` as the default.

So if you run this in a repo that was created with `master`, it renames it to `main` so it matches GitHub's naming.

### Example:

bash Copy Edit

git branch

Before:

markdown

\* master

Runc

bash

git branch -M main

After:

css

\* main

bash

git push <remote> <branch>

- \* `<remote>` -- usually `origin` (the name of your GitHub remote).
- \* `<branch>` -- the local branch you want to push.

So normally you'd write:

bash Copy Edit

git push origin main

git push -u origin main

- \* Pushes your local `main` branch → to the `origin` remote.
- \* The `-u` (or `--set-upstream`) option tells Git:  
"Remember this remote/branch pair, so in the future I can just do `git push` or `git pull` without typing `origin main` every time."

So after this, you can just type:

bash Copy Edit

git push

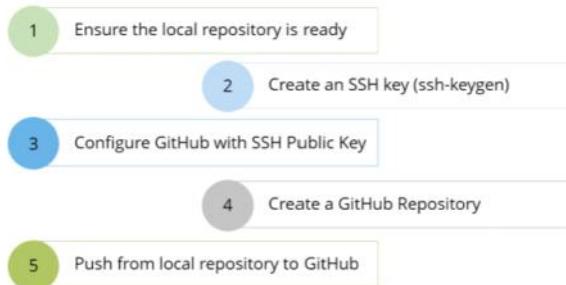
git pull

and Git knows it means `origin main`.

# Create Repositories (SSH Protocol)

Sunday, August 24, 2025 8:28 AM

## Steps to Create a Repository in GitHub Using SSH



### Initialize git:

```
$git init
```

### Stage all the files:

```
$git add .
```

### Commit:

```
$git commit -m "nothing to commit"
```

```
Git status
```

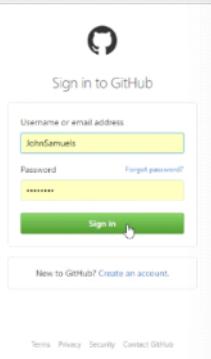
### get key for ssh:

```
ssh -keygen
```

```
00000801: INITIAL version
hp@DESKTOP-01792DU MINGW64 /d/dev/lgit/newRepossh (master)
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/hp/.ssh/id_rsa):
Created directory '/c/Users/hp/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/hp/.ssh/id_rsa.
Your public key has been saved in /c/Users/hp/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:dv6ylvauKulr3bud8XcdAHFNjFCLVi1DwN8k0ThGNC8 hp@DESKTOP-01792DU
The key's randomart image is:
+---[RSA 2048]---+
|B%o**.|
|.BX-o+|
|=E=o=|
+.+=.|
S . . o|
+ . . . o|
. . . .++|
. . . .++|
+--+ [SHA256]
```

Copy the location of the key.

### Go to github page and login:



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

The screenshot shows the GitHub account settings page. The left sidebar has a tree view of settings categories. The 'SSH and GPG keys' section is highlighted with a red box. The main area is titled 'Public profile' and contains fields for 'Profile picture', 'Name', 'Public email', 'Bio', 'URL', 'Company', and 'Location'. There are also sections for 'Blocked users', 'Repositories', 'Organizations', 'Saved replies', 'Authorized applications', and 'Installed integrations'.

## Add new SSH key to your account:

The screenshot shows the GitHub account settings page with the 'SSH and GPG keys' section highlighted. The 'SSH keys' section shows a message 'There are no SSH keys with access to your account.' It has a 'New SSH key' button and a text input field for 'Title' and 'Key'. Below it is a note about generating SSH keys. The right side of the screen shows the 'SSH keys' and 'GPG keys' sections, both of which are currently empty. A note at the bottom right says 'Learn how to generate a GPG key and add it to your account.'

## Create new repo in github: click on SSH protocol.

The screenshot shows a GitHub repository creation wizard for a 'SSH Remote Repo'. The top bar shows the repository path 'JohnSamuels / SshRemoteRepo'. The main area has tabs for 'Code', 'Issues', 'Pull requests', 'Projects', 'Wiki', 'Pulse', and 'Graphs'. A 'Quick setup' section provides instructions for setting up the repository on desktop or via HTTPS. It includes a command-line snippet for creating a new repository and another for pushing code from an existing repository. A note at the bottom says '...or import code from another repository' with a link to 'Request code'.

```
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:nThbg6KXUpJWg17ElIGCspRomTxdCARLviKw6E55V8.
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:

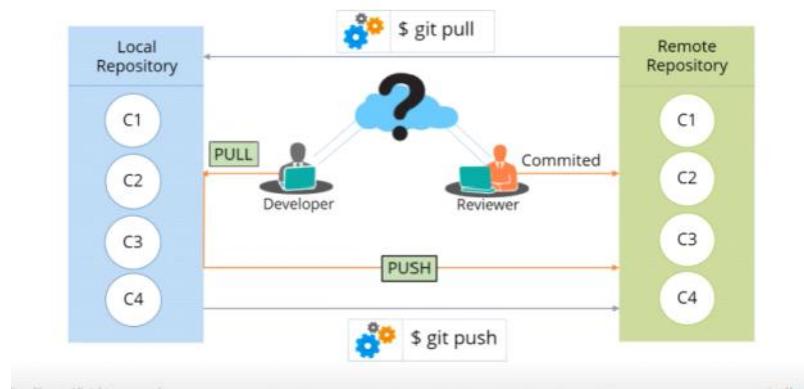
Add a file/ edit a file in the repo:

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
it
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
```

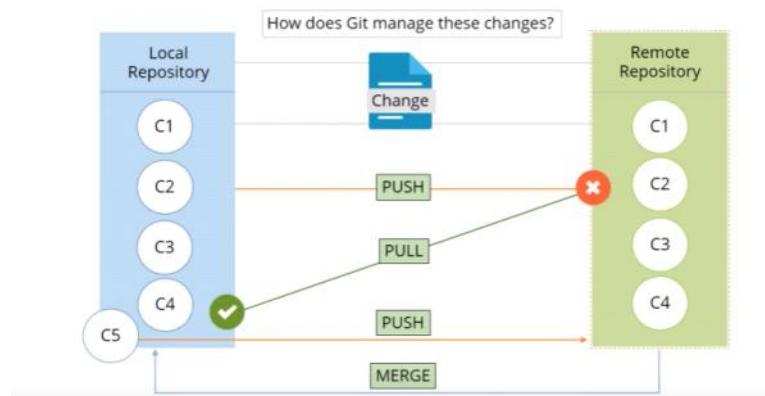
if globally: pull

```
$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.

### Managing Multiple Commits to a File



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|MERGING)
$ 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.git'
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|MERGING)
$ 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"
```

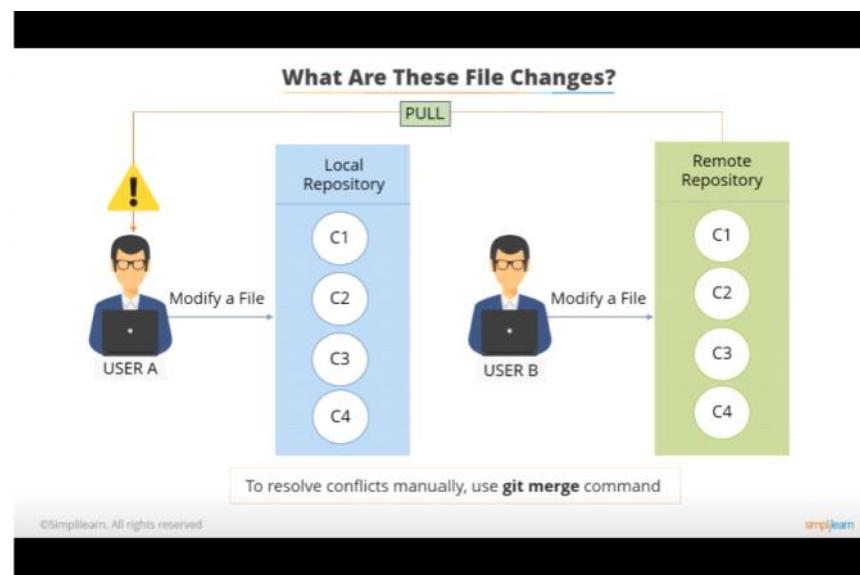
**Commit changes for pulled file:**

```
$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:**

```
ommor@Ommore5807 MINGW64 /c/Masters USA/Extra Projects/git_practice (main|MERG)
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.git'
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|MERG)
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"
```

**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

The screenshot shows the GitHub interface for a repository named "JohnSamuels / SshRemoteRepo". The top navigation bar includes links for Code, Issues (0), Pull requests (0), Projects (0), Wiki, Pulse, Graphs, and Settings. The main content area is titled "Labels" and shows a list of 7 labels: bug, duplicate, enhancement, help wanted, invalid, question, and wontfix. Each label entry includes a small icon, the label name, a count of "0 open issues", an "Edit" link, and a "Delete" link. A "New label" button is located at the top right of the labels list.

## ◊ 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).