# GITHUB and LINUX

**GIT HUB INSTALLATION**

On **Linux (CentOS/RedHat)**:

sudo yum install git -y

On **Ubuntu/Debian**:

sudo apt update

sudo apt install git -y

**On Windows:**

* Download Git installer from <https://git-scm.com/download/win> and follow the setup wizard.

**On macOS:**

brew install git

(if you have Homebrew installed)

**EMPTY FILE CREATION**

touch filename.txt

touch myfile.txt

**TEXT EDITORS**

echo "Hello, Linux!" > myfile.txt

vi myfile.txt

nano myfile.txt

cat > myfile.txt

**READ A FILE**

cat filename.txt

**SOME BASIC COMMANDS**

**🔹 1. pwd — Show current directory**

pwd

✅ Tells you where you are in the filesystem.

**🔹 2. ls — List files and folders**

ls

ls -l # detailed list with permissions

ls -a # show hidden files like `.git`

**🔹 3. cd — Change directory**

cd myproject/ # go into a folder

cd .. # go up one level

**🔹 4. mkdir — Create a new directory**

mkdir myproject

**🔹 5. touch — Create a new (empty) file**

touch file.txt

**🔹 6. echo — Create or append content to a file**

echo "Hello Git" > file.txt # overwrite

echo "More content" >> file.txt # append

**🔹 7. cat — View file contents**

cat file.txt

**🔹 8. nano or vi — Edit a file (command-line editors)**

nano file.txt

vi file.txt

**🔹 9. rm — Delete a file**

rm file.txt

**🔹 10. chmod** — chmod stands for "change mode". It’s used to change the permissions of a file or directory.

**Read (r) 4 | User 7 –>rwx**

**Write(w) 2 | Group 5 –>r-x**

**Exec(x) 1 | Others 5 🡪r-x**

A screenshot of a computer

AI-generated content may be incorrect.

Once you create a new file, you need to commit it. If the file is newly created, git status will show it as an untracked file. If you modify an existing file, it will show as a modified file. Below are the steps to handle both cases.

**Untracked files**

git add example.txt/ add .

git commit -m "Add new file example.txt"

git push origin main

**Modified files**

git commit -am "Update existingfile.txt"

git push origin main

A computer screen shot of a computer code

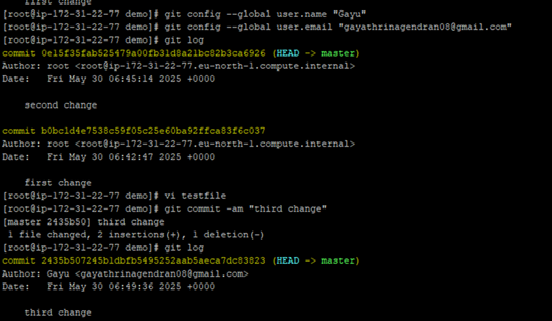
AI-generated content may be incorrect.

To change the name of the root user :

git config --global user.name "Your New Name"

git config --global user.email "yourname@example.com"

These commands update the name and email Git uses for all repositories on the system.



GitHub **requires a Personal Access Token (PAT)** instead of a password for Git operations like git push, git pull, and git clone **over HTTPS**. This is for **security reasons**.

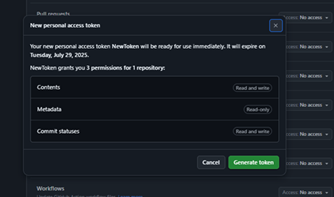
1. **Generate a Token** from GitHub:

* Go to: <https://github.com/settings/tokens>
* Click **"Generate new token"** (or "Fine-grained token" for more control)
* Select scopes like:
  + repo (for repo access)
  + workflow, gist, etc., depending on what you need
* Copy the token (you won’t be able to see it again)

A screenshot of a computer

AI-generated content may be incorrect.

Create a name 🡪select the Repository 🡪give permissions



**ADD A GITHUB REPOSITORY USING GIT REMOTE**

**📍 Step 1: Initialize Git (if not already initialized)**

git init

**📍 Step 2: Add a remote repository**

Get your GitHub repository URL. It looks like this (HTTPS format):

<https://github.com/yourusername/your-repo.git>

A screenshot of a computer

AI-generated content may be incorrect.

**Now run:**

**git remote add origin https://github.com/yourusername/your-repo.git**

**🔹** origin is the name for the remote (default name).

**📍 Step 3: Verify the remote is added**

**git remote -v**

origin https://github.com/yourusername/your-repo.git (fetch)

origin https://github.com/yourusername/your-repo.git (push)

**📍 Step 5: Push to GitHub**

git push -u origin main

If your branch is called master, use origin master instead.

A computer screen shot of a computer program

AI-generated content may be incorrect.

**📍 Step 5: View in GitHub**

**A screenshot of a computer

AI-generated content may be incorrect.**

**BRANCHES**

Once you commited you can automatically the master branch will be created

**✅ Step 1: Check the current branch**

git branch🡪To view the branch

Output might look like:

markdown

\* master

The asterisk \* shows the branch you're currently on.

**🌿 Step 2: Create a new branch**

To create a new branch (e.g., dev):

git branch dev

This creates a new branch but does not switch to it yet.

**🔄 Step 3: Switch to the new branch**

git checkout dev

You are now working in the dev branch. Any commits you make will stay on this branch.

✅ Alternatively, you can do both at once:

git checkout -b dev

That means:

* -b = create a new branch
* checkout = switch to it

**🚀 Step 4: Push the new branch to GitHub**

Once you've committed something on the new branch:

git push -u origin dev

* This creates the dev branch on GitHub
* -u sets upstream so future pushes can just be git push

## **📋 Step 5: View all branches**

To see **remote branches and local branch**:

git branch -r , git branch -a

A screenshot of a computer program

AI-generated content may be incorrect.

**CONFLICT**

A **GitHub conflict** (more accurately, a **Git merge conflict**) happens when Git cannot automatically merge changes between branches because the same part of a file was changed differently in two branches.

**🛠️ Git Merge Tool**

A **merge tool** is a utility that helps you **visually resolve merge conflicts** in Git. It shows conflicting changes side-by-side and allows you to select or edit how the final merged file should look.

A screen shot of a computer screen

AI-generated content may be incorrect.

 **Set Vimdiff as the merge tool:**

git config --global merge.tool vimdiff

* Uses vimdiff for visualizing and resolving merge conflicts.

 **Enable 3-way merge view:**

git config --global merge.conflictstyle diff3

* Adds a ||||||| BASE section to conflicts, showing the common ancestor.

 **Suppress confirmation prompt:**

git config --global mergetool.prompt false

* Automatically opens vimdiff without asking which tool to use for each file.

 **Launch the merge tool:**

git mergetool

* Opens all conflicted files in Vimdiff to resolve them.

**A screenshot of a computer

AI-generated content may be incorrect.**

**Commit the Merge**

Once all conflicts are resolved and added:

git commit

**REBASE**

git rebase is a powerful Git command that moves or reapplies commits from one branch onto another to make the history linear and cleaner.

A computer screen with text and numbers

AI-generated content may be incorrect.

**Rebase** takes the commits from your current branch and "replays" them on top of another branch.

**📌 Common Use Case**

git checkout feature-branch

git rebase main

This means:

Take the changes made in feature-branch and reapply them **on top of** the latest commits in main.

**CHERRY PICK**

You made a useful commit on dev, and now you want that specific commit on main — without merging the whole dev branch.

**1. ✅ Check commit log in dev**

Switch to the dev branch and find the commit you want:

git checkout dev

git log --oneline

**2. 🌿 Switch to the target branch (main/master)**

git checkout main/master

**3. 🍒 Cherry-pick the commit**

git cherry-pick <commit-hash>

**A screenshot of a computer program

AI-generated content may be incorrect.**