Practical Git & GitHub Guide (No Theory)

Prerequisites

1. Install Git
2. Create a GitHub Account

Step 1: Create a GitHub Repository (Before Pushing Code)

(Do this first if you don't have a repo on GitHub yet!)

Step 1.1: Go to GitHub

- Open github.com → Log in → Click **"+"** → **"New repository"**

Step 1.2: Fill Repository Details

- **Repository name**: Your-Repo-Name
- **Description**: (Optional)
- **Public/Private**: Choose visibility
- **Initialize README?**: ✅ *(Recommended for first-timers)*
- **Add .gitignore?**: Optional (e.g., Python, Node)
- **License?**: Optional

Step 1.3: Click "Create Repository"

- Now, you have a GitHub repo! Copy the **HTTPS URL** (e.g., **https://github.com/YourUsername/Your-Repo-Name.git)**

Practical Git & GitHub Guide (No Theory)

Step 2: Pushing Files to GitHub Repository

Step 2. 1: Create a Folder

Create a new folder on your computer where you want to store your project files.

Step 2.2: Open Git Bash in the Folder

Right-click inside the folder → "Show more options" → "Git Bash Here" (Windows)

(Mac/Linux users can open Terminal and navigate to the folder using cd)

Step 2.3: Check if it's a Git Repo (Optional)

ls -a

If you see a .git folder, it's already a Git repo. If not, proceed to Step 4.

Step 2.4: Initialize Git Repository

git init

Converts your folder into a Git repository.

Step 2.5: Set Your GitHub Username

git config --global user.name "YourGitHubUsername"

Links your commits to your GitHub account.

Step 2.6: Set Your GitHub Email

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

Must match the email registered on GitHub.

Practical Git & GitHub Guide (No Theory)

Step 2.7: Check File Status

git status

Shows untracked/modified files (marked in red).

Step 2.8: Add Files to Staging Area

git add . (it is dot)

.(dot) adds all files. You can also use git add filename for specific files.

Step 2.9: Check Status Again

git status

Files now appear in green (staged and ready to commit).

Step 2.10: Commit Changes

git commit -m "Your commit message here"

Saves changes with a description.

Step 2.11: Link to GitHub Repository

git remote add origin https://github.com/YourUsername/YourRepo.git

Replace the URL with your GitHub repo's HTTPS link.

Step 2.12: Push to GitHub

git push origin main

main is the default branch. Use master if your repo uses it.

Practical Git & GitHub Guide (No Theory)

Step 3: Working with Branches

Step 3.1: Check Current Branch

git branch

Shows all branches (* marks the current one).

Step 3.2: Create a New Branch

git branch new-branch-name

Creates a new branch but does not switch to it.

Step 3.3: Switch to the New Branch

git checkout new-branch-name

Moves you to the new branch.

(Alternatively, use git switch new-branch-name in newer Git versions.)

Step 3.4: Push the New Branch to GitHub

git push origin new-branch-name

Step 4: Additional Useful Commands

Step 4.1: View Commit History

git log

Displays a list of all previous commits in the current repository. It shows commit IDs, authors, dates, and messages. Useful for tracking changes.

Practical Git & GitHub Guide (No Theory)

Step 4.2: Pull Latest Changes from GitHub

git pull origin main
Fetches and merges the latest updates from the "main" branch of the remote GitHub repository into your local repository.

Step 4.3: Merge a Branch into the Current Branch

git merge branch-name
Combines the changes from the specified branch (branch-name) into your currently checked-out branch.

Step 4.4: Clone a GitHub Repository Locally

git clone repo-url
Downloads the entire content of a GitHub repository (including history) to your computer. You get a full local copy of the project.

Practical Git & GitHub Guide (No Theory)

Merging Branches on GitHub (UI)

Definition:
Merging branches means combining changes from one branch (e.g., feature branch) into another (e.g., main branch). This helps to bring new code or features into the main project.

Steps:

Step 1: Go to your GitHub repository
Open the repository where you created the branches.

Step 2: Click on the "**Pull requests**" tab
Located at the top of the repo, next to Issues and Code.

Step 3: Click the "**New pull request**" button
This starts the merge process between two branches.

Step 4: Choose branches to compare

- Base branch → the branch you want to merge **into** (usually `main`)
- Compare branch → the branch you want to merge **from** (like `feature-1`)

Step 5: Review the changes
You'll see file changes and commits that will be merged.

Step 6: Click "**Create pull request**"
Give a title and description for the changes.

Step 7: Click "**Merge pull request**"
Once ready (or reviewed), click to start the merge.

Step 8: Click "**Confirm merge**"
This completes the merge.

Step 9 (Optional): Delete the feature branch
Click "**Delete branch**" if you no longer need the merged branch.

Forking a Repository on GitHub (UI)

Definition:

Forking a repository creates a personal copy of someone else's GitHub repository under your own account. You can freely make changes without affecting the original project.

Steps:

Step 1: Open the original GitHub repository
Go to the repo you want to copy (fork).

Step 2: Click the "**Fork**" button
Located in the top-right corner of the page.

Step 3: Select your GitHub account
Choose where you want the fork to be created (usually your own account).

Step 4: GitHub creates a fork
It copies the original repository to your account.

Step 5 (Optional): Rename your forked repo
Click the **Settings** tab in your new repo if you want to change its name.

Step 6: Work on your fork
Now you can edit code, create branches, and make changes independently.

Step 7 (Optional): Contribute back
After making changes, you can open a pull request to suggest those changes to the original repo.