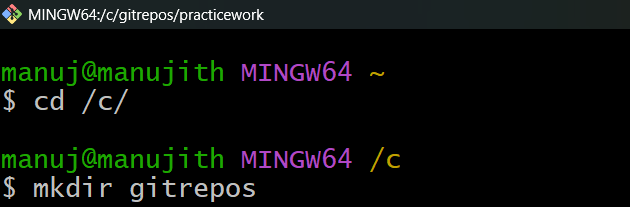
Here’s a step-by-step note I created while learning Git basics and version control. This document covers initialization, committing, branching, and pushing changes to GitHub in simple words.

Versioning



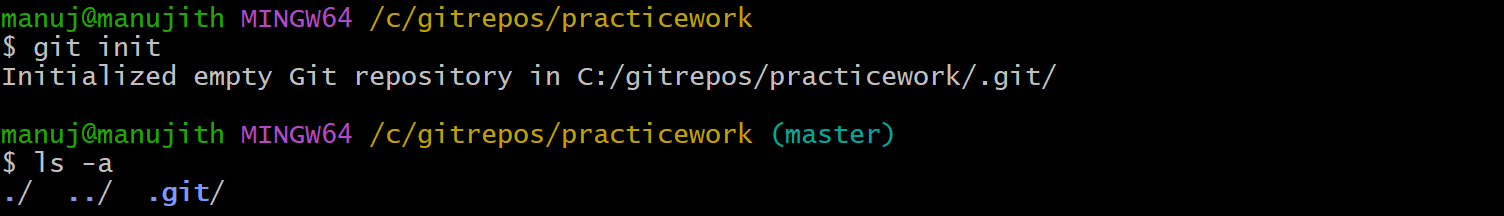
* **cd** = change directory.
* /c/ means you’re moving into the **C drive** of your Windows system (because Git Bash shows Windows drives as /c/, /d/, etc.)
* 👉 Purpose: You moved from your **home directory (~)** into the **C drive**
* **mkdir** = make directory (create folder).
* gitrepos is the name of the folder you are creating.  
  👉 Purpose: You created a new folder named **gitrepos** inside your **C drive**

A black screen with yellow and purple text

AI-generated content may be incorrect.

* **mkdir** = make directory (create a new folder).
* **practicework** = the name of the new folder you are creating.

👉 **Purpose:**  
Inside your **gitrepos** folder, you just created a new folder called **practicework**



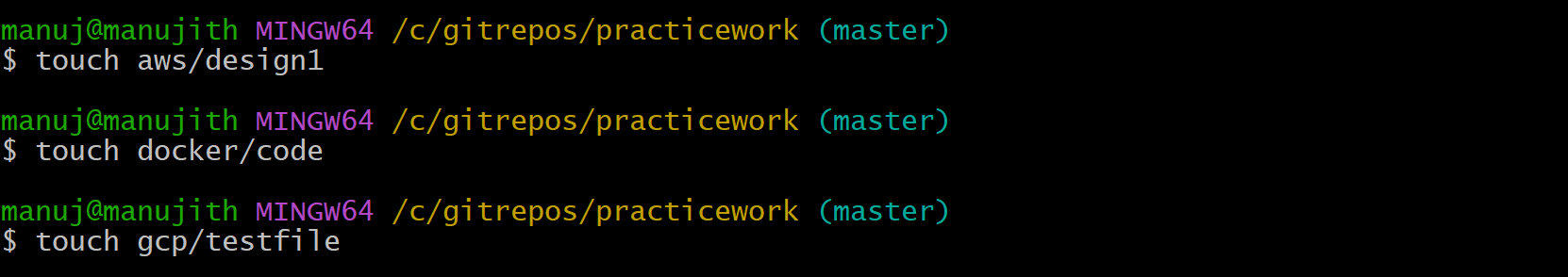
Cmd: manuj@manujith MINGW64 /c/gitrepos/practicework **(master**)

Master :  your current working branch.

* Every new commit you make will go into this branch, until you create or switch to another branch.
* Initializes a **new Git repository** in your current folder.
* It creates a hidden folder called **.git/** → this is where Git stores all the history, commits, and configuration for your repo.

A screen shot of a computer

AI-generated content may be incorrect.

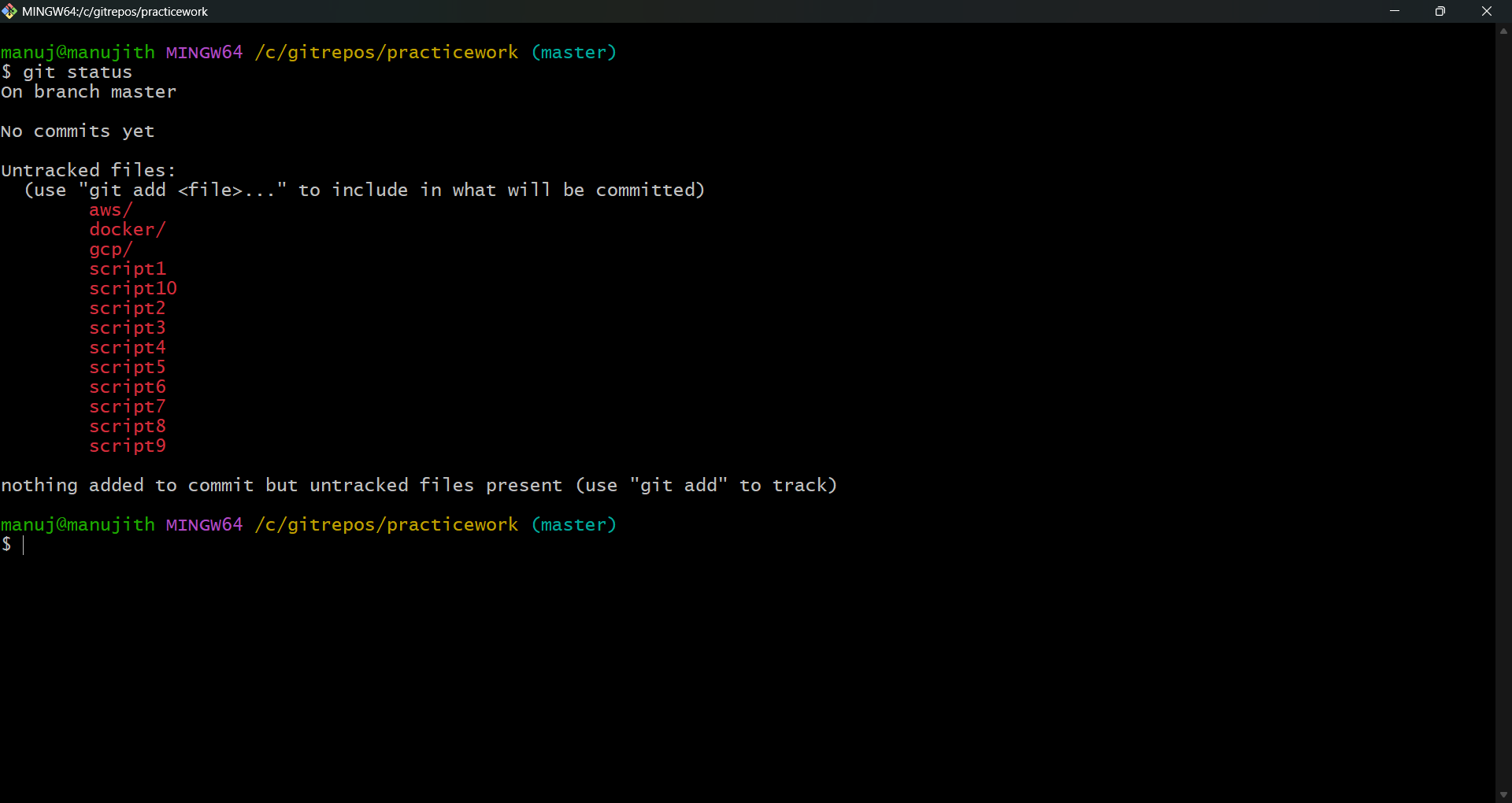
* **touch** = creates empty files.
* {1..10} = Bash **brace expansion** → automatically expands into numbers 1 through 10.  
  👉 So it creates **10 files** at once

**📌 Empty Folders in Git – Important Note**

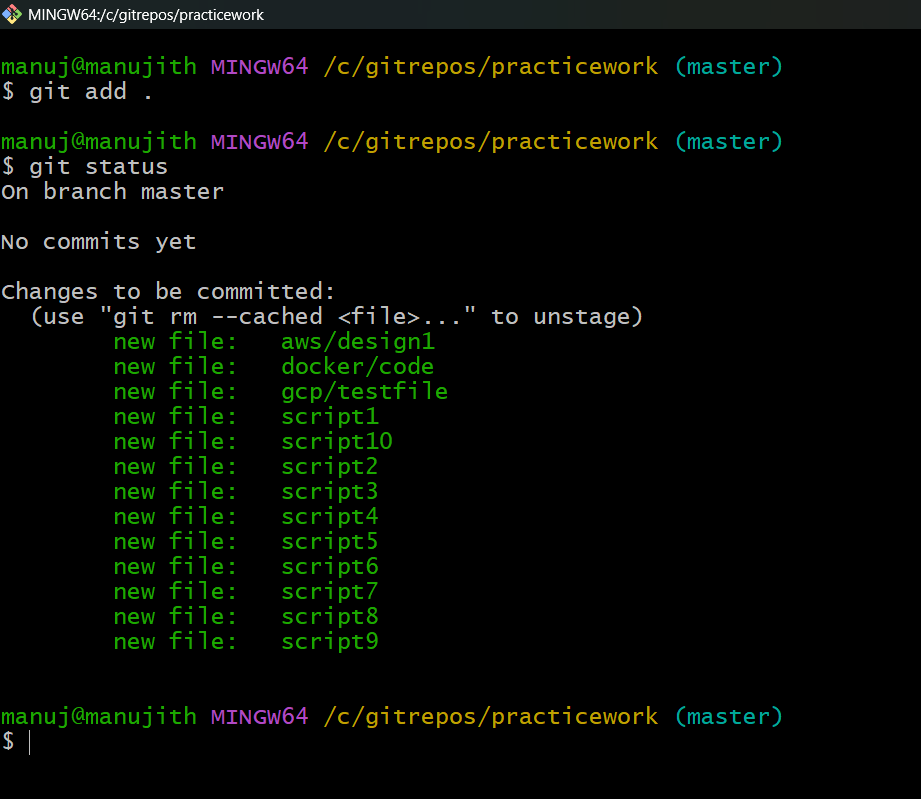
* Git does **not track empty folders**.
* Only **files** are tracked (changes, versions, commits).
* If a folder is empty, it will be ignored when pushing to GitHub.

**✅ Solution: Use a placeholder file**

To keep an empty folder in Git, developers usually add a dummy file (often called. gitkeep)



git status shows the current state of your repository. In this case, all files and folders are **untracked**, meaning Git sees them but isn’t tracking changes until we add them with git add .



* Before: files were **untracked** (Git saw them but ignored them).
* After git add .: files are now **staged**, waiting to be committed into history

A **commit** is like taking a **snapshot** of your project.  
👉 Now Git has officially saved the state of all your files at this point in history.

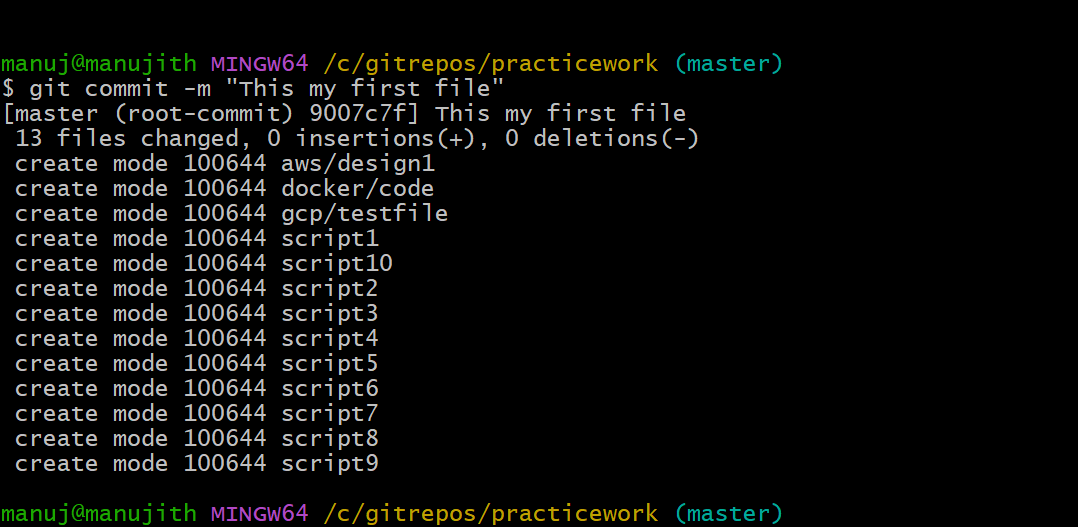
git config --global user.name "YourNewUsername"

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

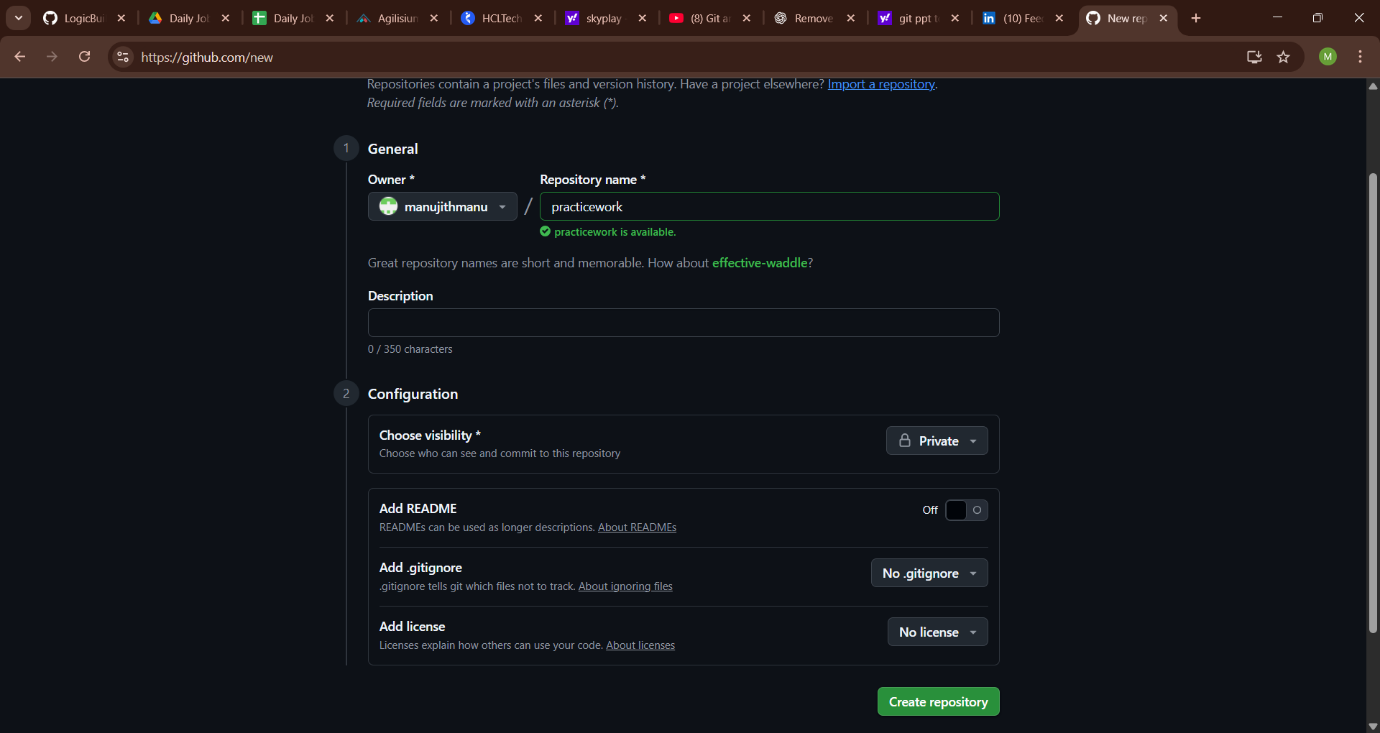
**🔹 What it does**

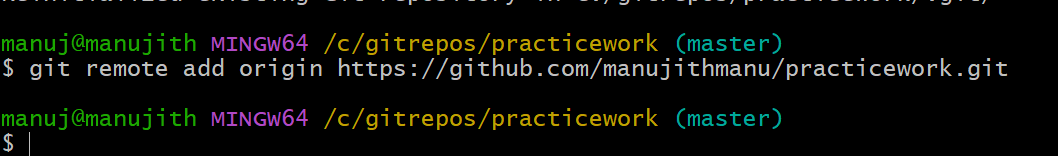
* **git config** → Used to configure Git settings.
* **--global** → Means the setting applies to **all repositories** on your computer.
* **user.name** → Sets the name that will appear in your commits.
* **user.email** → Sets the email that will appear in your commits.

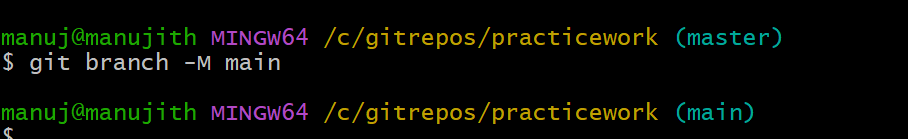
👉 This is **not logging into GitHub**.  
It only tells Git **what name and email to record in commit history**.



* **git commit** → Saves your changes in the Git **local repository history** (it does **not** go to GitHub yet).
* **-m** → Stands for **message**. You use it to describe what this commit is about.
* "This is my first file" → The commit message. It should explain the purpose of the changes you made.
* After creating a local Git repository, you can create a new repository in GitHub. The repository name does **not need to be the same** as your local folder.
* What matters is setting the correct remote URL using git remote add origin <GitHubRepoURL>.
* If the URL is set properly, you will be able to connect and push your local commits to any repository name on GitHub.



This command → **Connects your local repo to a remote GitHub repo (named origin).**



**One-line meaning**

This command → **Renames your current branch to main.**

**🔹 Why**

* Old Git default branch name = **master**
* New GitHub default branch name = **main**
* To match GitHub, we rename it with -M.

✅ Now your active branch is main.

**What the output means (in simple words):**

* **Enumerating / Counting / Compressing / Writing objects** → Git is preparing your files and sending them to GitHub.
* **Total 6 (delta 0)** → 6 objects (files/history) were uploaded.
* **To** [**https://github.com/**](https://github.com/?utm_source=chatgpt.com)**...** → Shows the remote repo URL where your code went.
* **[new branch] main → main** → A new branch main was created on GitHub from your local main.
* **branch 'main' set up to track 'origin/main'** → Now your local branch and GitHub branch are connected (tracking each other).

**🔹 One-line summary**

Your **local main branch is now uploaded to GitHub** and linked with the remote main branch.

**A black background with a black screen

AI-generated content may be incorrect.**

**What the output means (in simple words):**

* **Enumerating / Counting / Compressing / Writing objects** → Git is preparing your files and sending them to GitHub.
* **Total 6 (delta 0)** → 6 objects (files/history) were uploaded.
* **To** [**https://github.com/**](https://github.com/?utm_source=chatgpt.com)**...** → Shows the remote repo URL where your code went.
* **[new branch] main → main** → A new branch main was created on GitHub from your local main.
* **branch 'main' set up to track 'origin/main'** → Now your local branch and GitHub branch are connected (tracking each other).

**Simple term:**

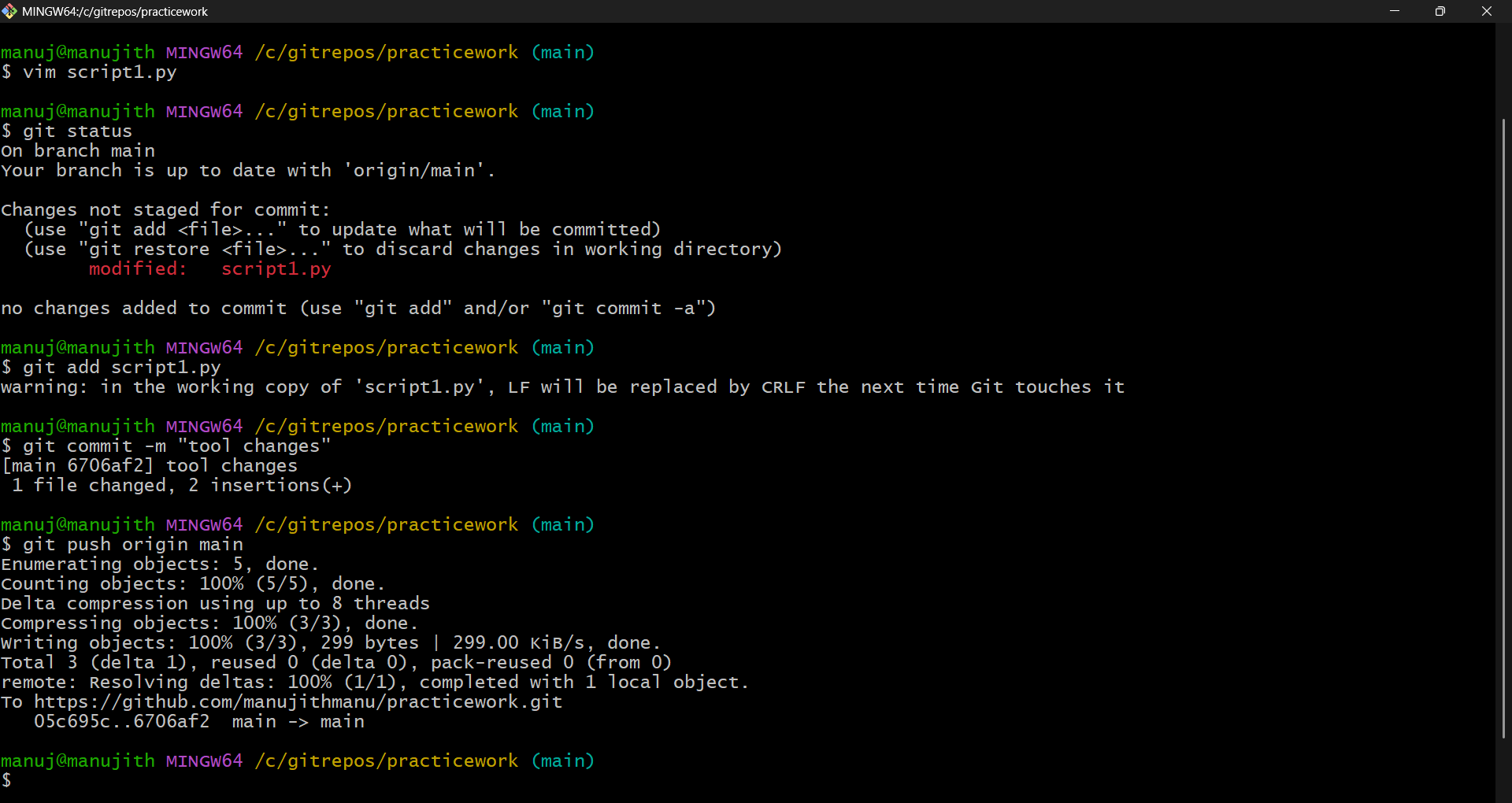
Your **local main branch is now uploaded to GitHub** and linked with the remote main branch.

A screenshot of a computer

AI-generated content may be incorrect.

This confirms that:

* You **committed locally** (with git commit).
* You **pushed successfully** (with git push -u origin main).
* GitHub is now showing the commit history + files you uploaded.



**1. Edit a file**

You opened and edited script1.py:

vim script1.py

**2. Check status of changes**

git status

* Git showed modified: script1.py, meaning the file was changed but not staged yet.

**3. Stage the file**

git add script1.py

* This moves the file into the **staging area** (ready for commit).

**4. Commit the change with a message**

git commit -m "Tool changes"

* Saves the staged file into local Git history with the message "Tool changes".

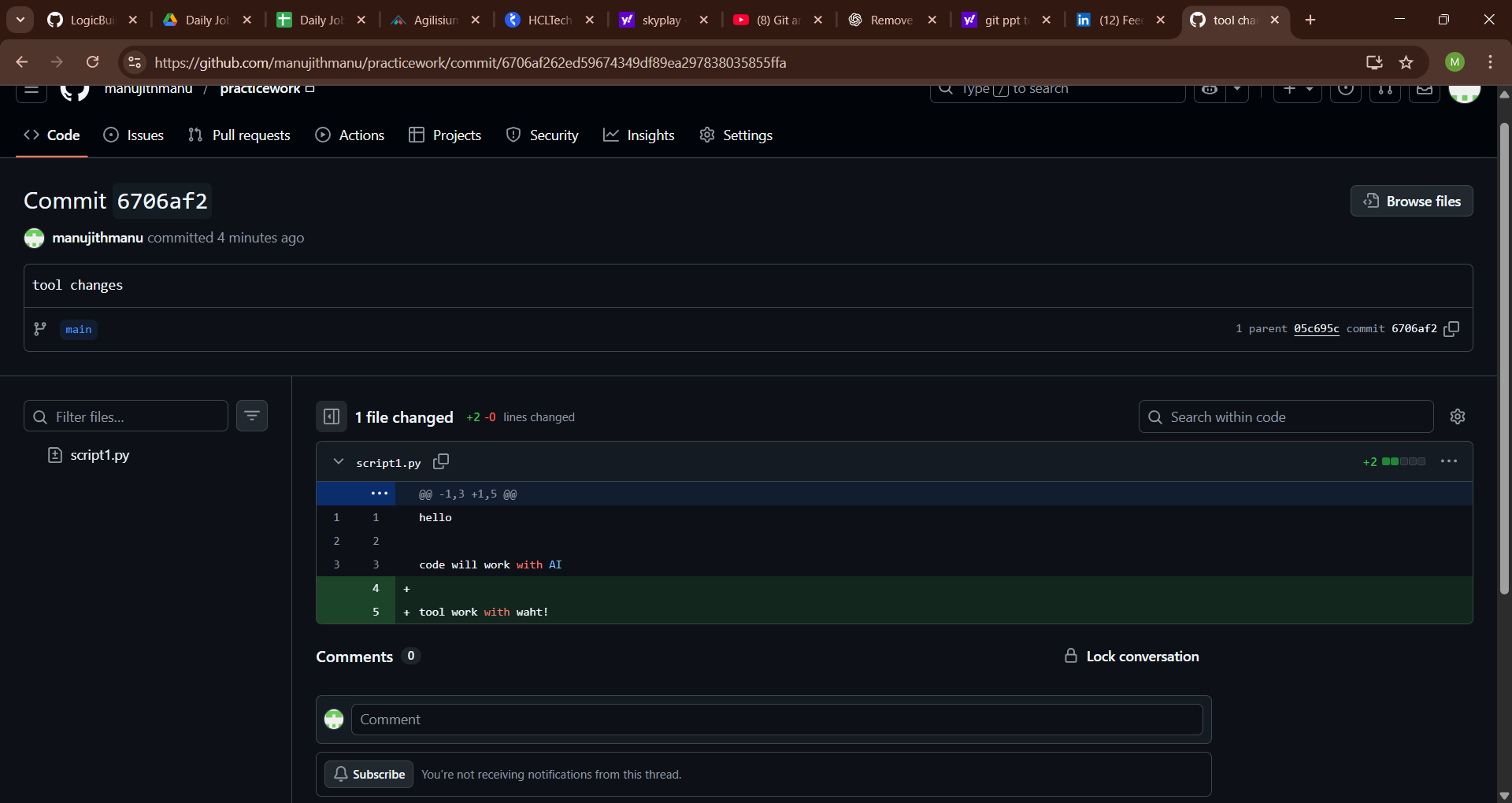
**5. Push changes to GitHub**

git push origin main

* Uploads the commit from your local main branch to the remote main branch on GitHub.

**🔹 One-line meaning for the cycle**

👉 **Edit → git add → git commit → git push = update GitHub with your latest changes.**



* Tracking changes and modifications
* In Git, green text means new lines were added, and red means lines were removed.