1. **Git-HOL**

**LAB 4**

## ****Why conflicts happen****

A **merge conflict** occurs when:

Git tries to merge two branches

The **same part** of a file is modified differently in both branches

Git **can’t decide automatically** which change to keep

Example scenario:

Branch **main** has:

int x = 10;

Branch **feature** has:

int x = 20;

When merging, Git doesn’t know whether 10 or 20 should be kept.

## ****2. How to detect a merge conflict****

When we run:

git merge branch\_name

If there’s a conflict, Git will show:

CONFLICT (content): Merge conflict in file.txt

Automatic merge failed; fix conflicts and then commit the result.

## ****3. How Git marks conflicts in the file****

Open the conflicted file, you’ll see:

<<<<<<< HEAD

int x = 10;

=======

int x = 20;

>>>>>>> branch\_name

<<<<<<< HEAD → Your current branch’s version (the branch you are merging into)

======= → Separator between versions

>>>>>>> branch\_name → Incoming branch’s version

## ****4. Steps to resolve a conflict****

**Open the conflicted file(s)**:

git status # shows which files have conflicts

**Decide which changes to keep**:

Keep your version

Keep their version

Combine both

**Edit the file** and remove the conflict markers (<<<<<<<, =======, >>>>>>>).

**Mark the conflict as resolved**:

git add file.txt

**Complete the merge**:

git commit

(This commit will have a default merge message)

## ****5. Using merge tools****

Instead of editing manually, you can use GUI merge tools:

**VS Code** → Shows “Accept Current Change” / “Accept Incoming Change”

**P4Merge**, **Meld**, **Beyond Compare** → Dedicated merge conflict UIs

git mergetool

## ****6. Preventing future conflicts****

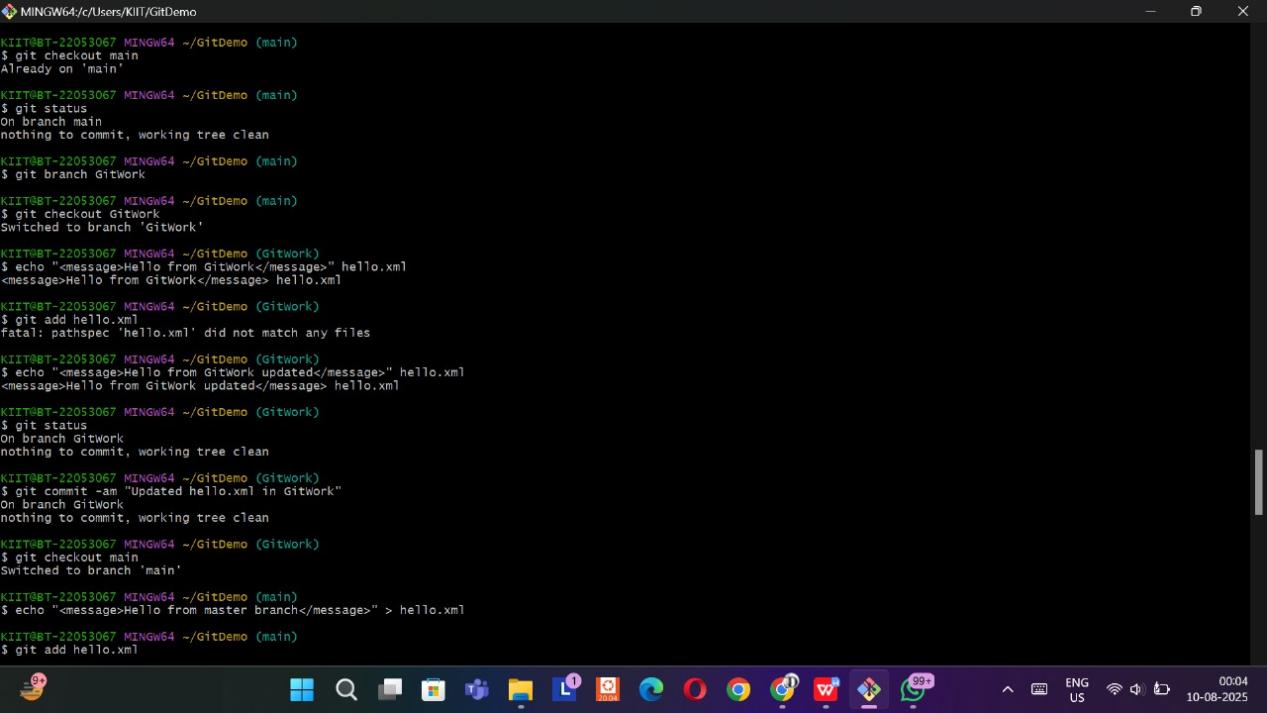
Pull frequently from the main branch:

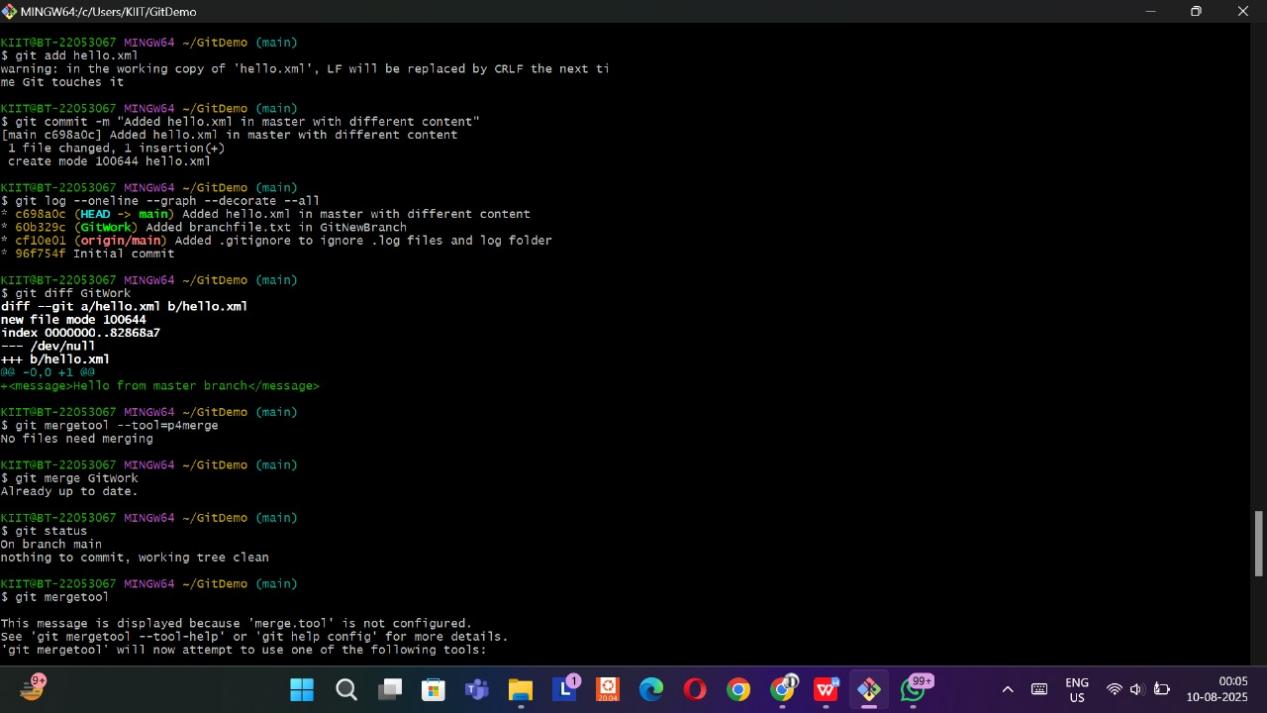
git pull origin main

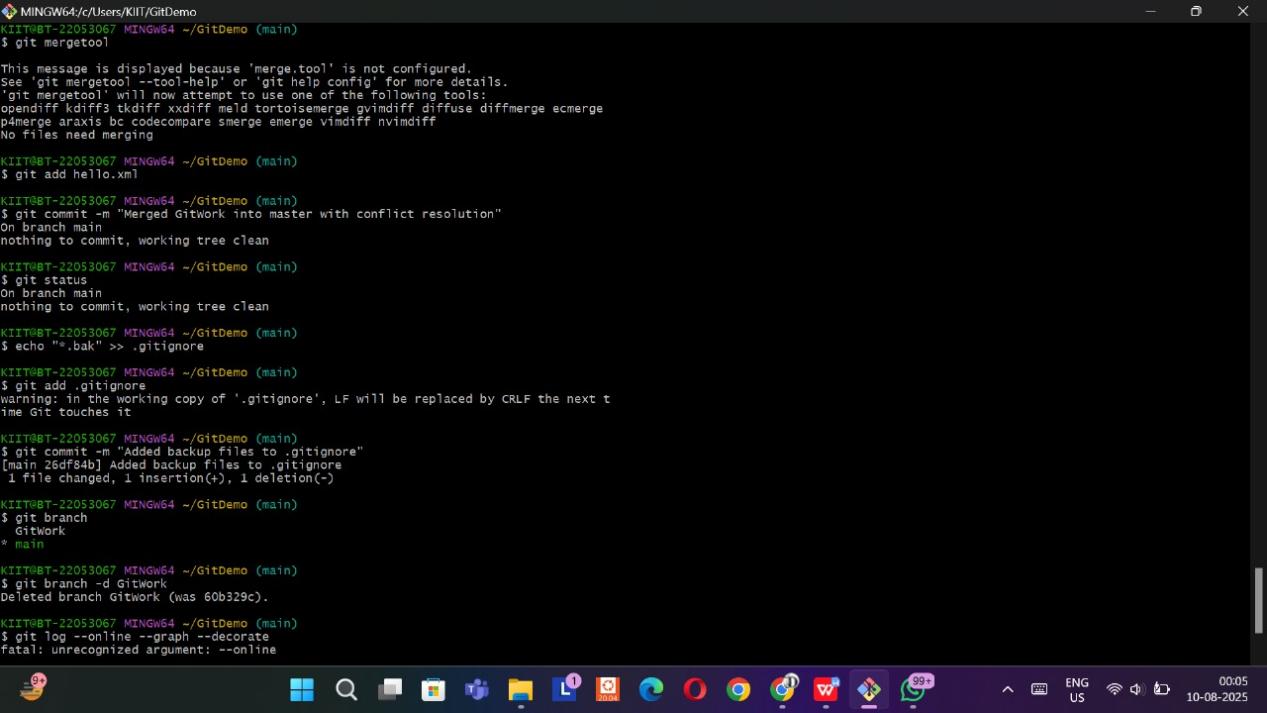
Keep your changes small and focused

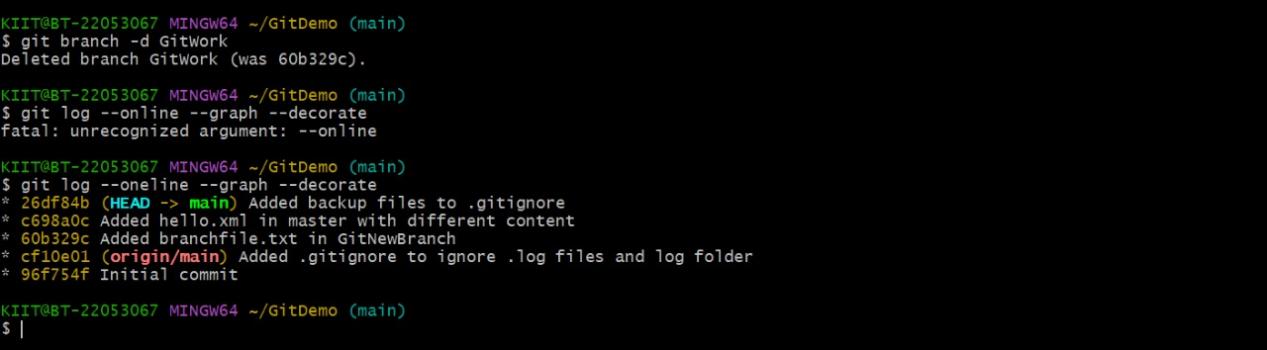
Communicate with teammates about files being modified

**Output:**

****

****

****

****