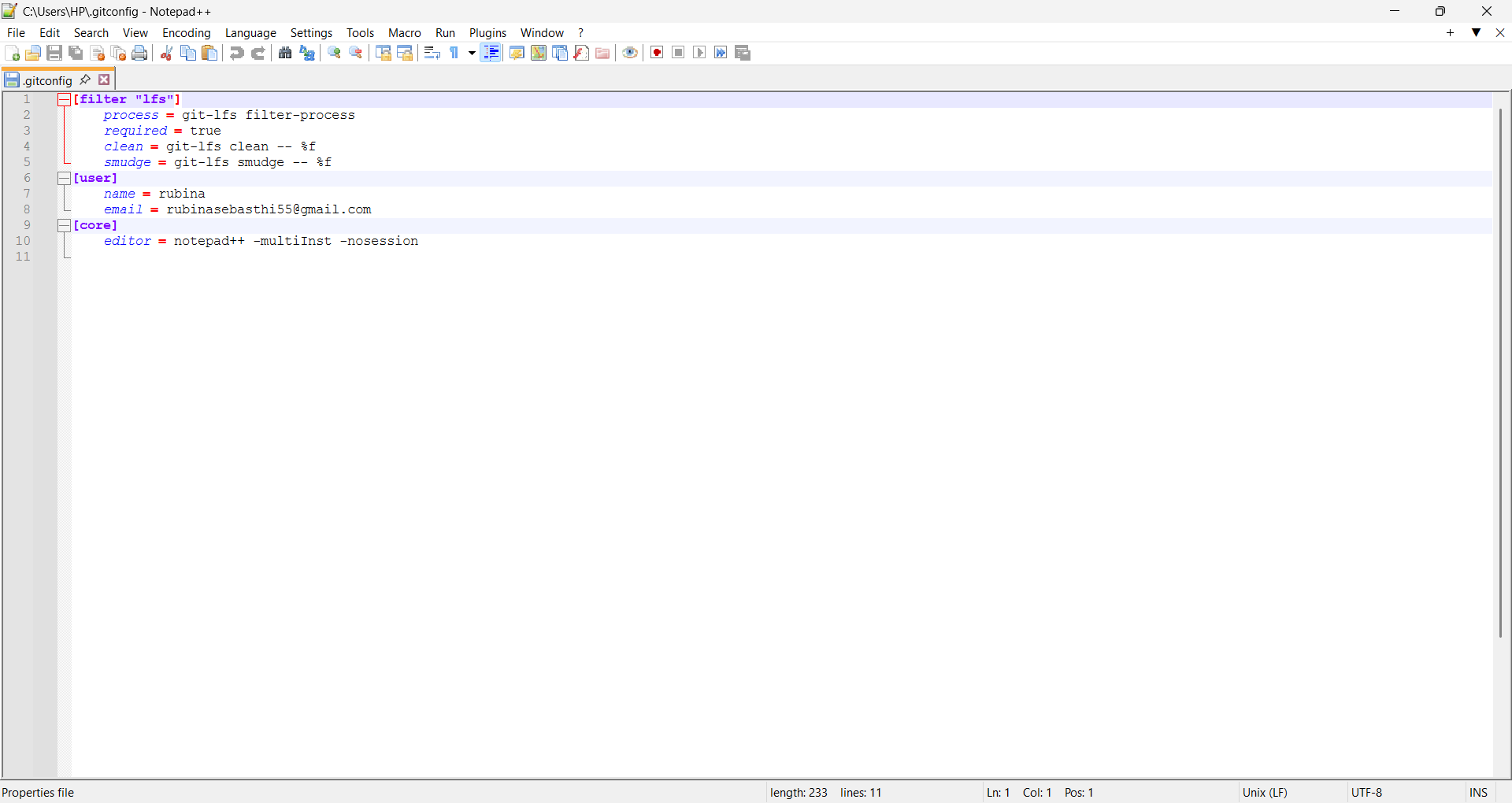
**Week-8 GIT**

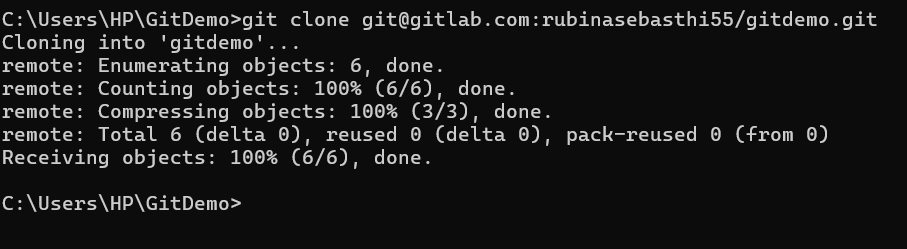
**SupersetID:6383615**

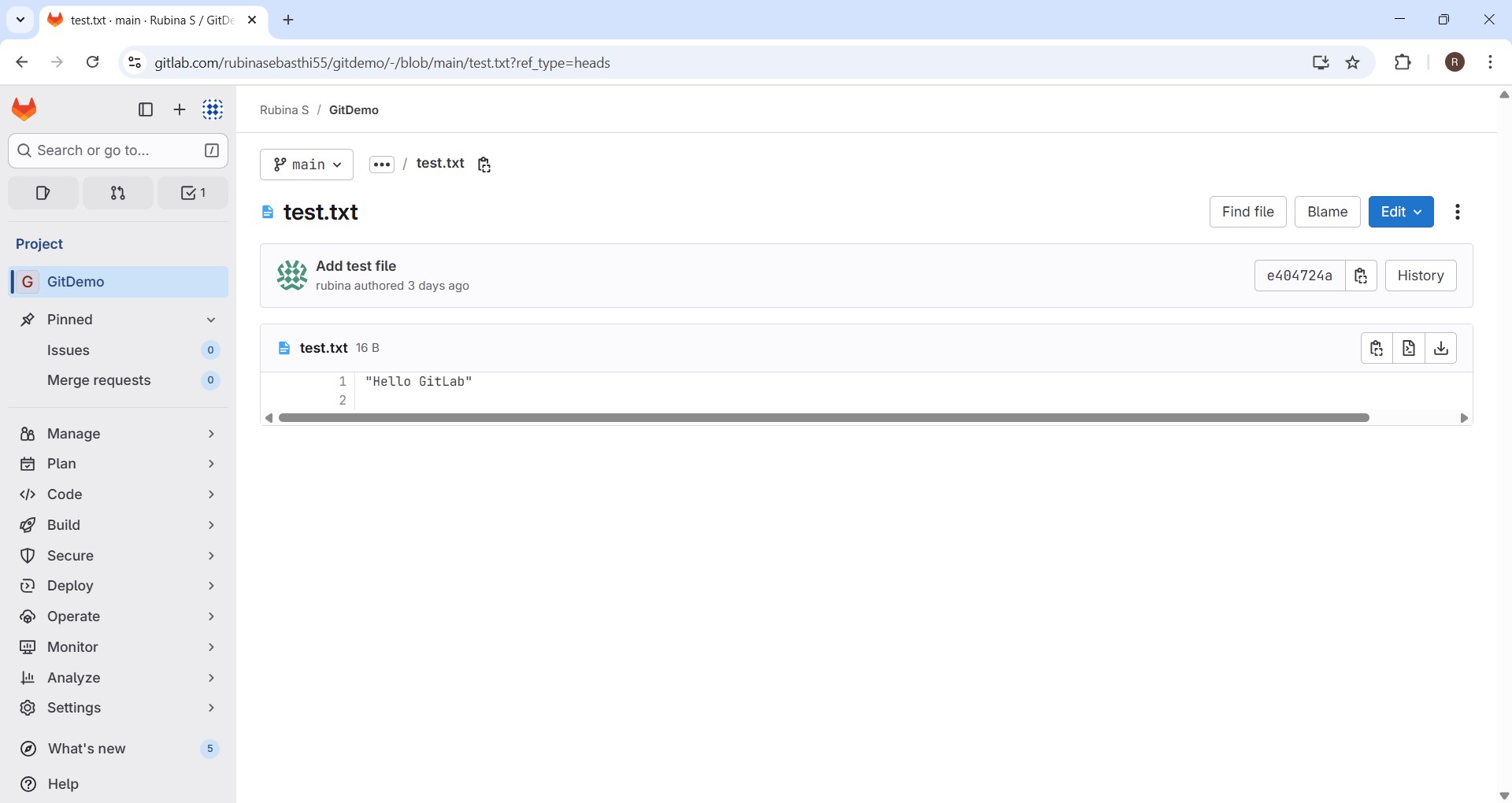
**Rubina S**

# GIT-HOL

**Output**

****

****



# GIT-HOL

**Objectives**

* + Explain git ignore
  + Explain how to ignore unwanted files using git ignore

**Answers**

**Explain .gitignore**

1. .gitignore is a configuration file in Git that specifies which files or directories should be ignored by Git (i.e., not tracked or committed).
2. It helps keep the repository clean by excluding temporary, build, log, or system files that are not relevant to the project’s source code.

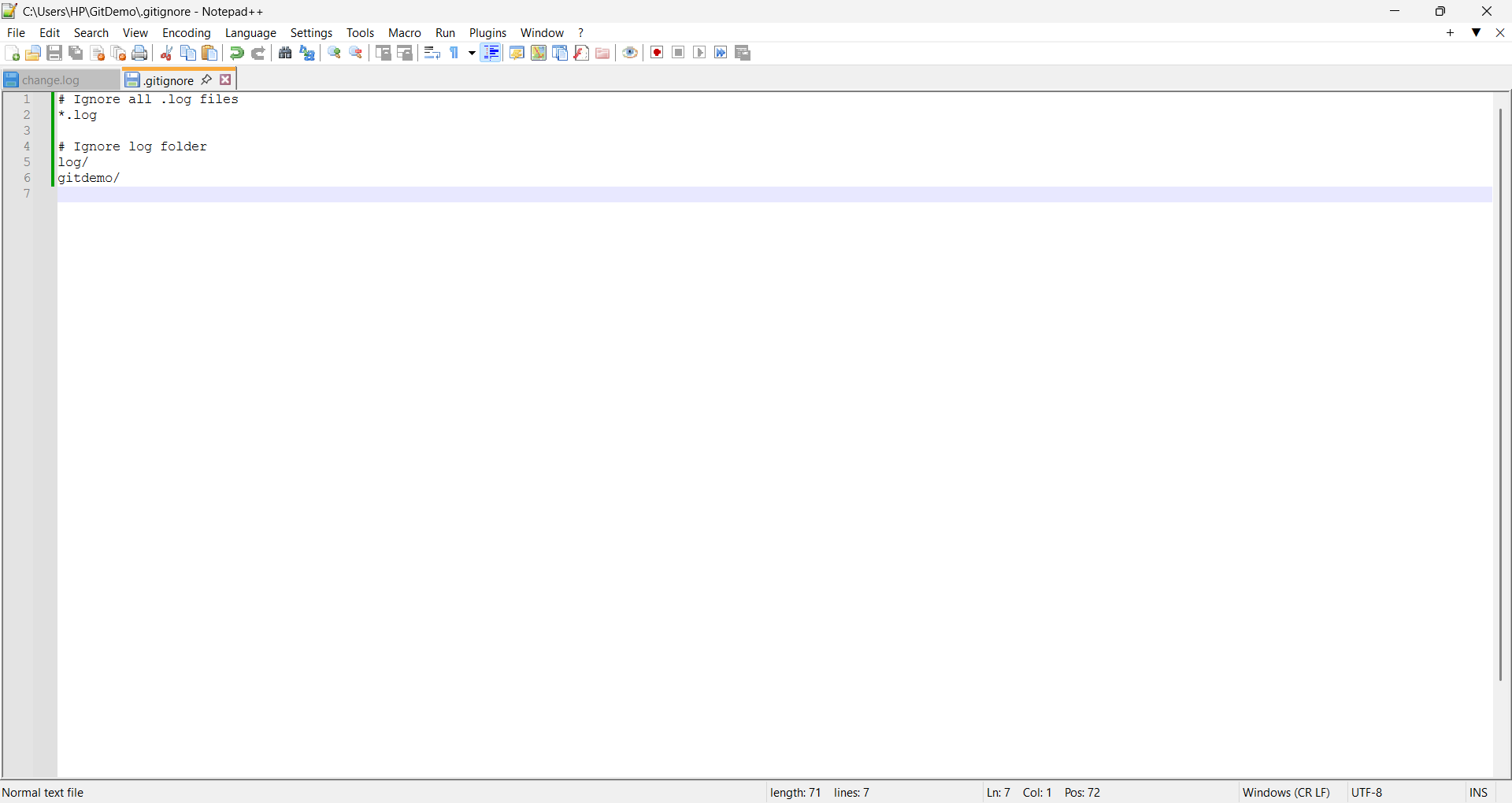
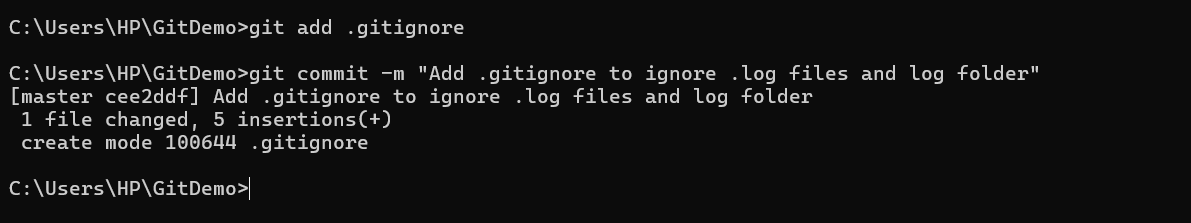
**Explain how to ignore unwanted files using .gitignore**

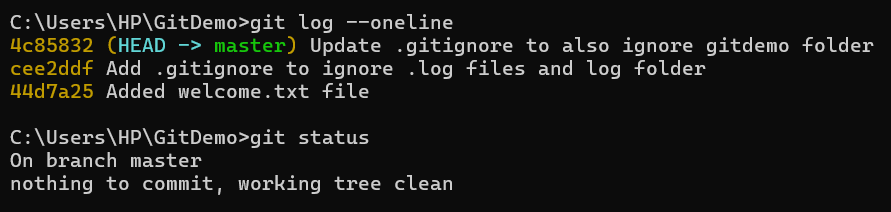
1. Add file patterns (like \*.log for log files or log/ for a folder) inside the

.gitignore file to tell Git to exclude them from tracking.

1. Once added to .gitignore, these files will remain in your working directory but won’t appear in git status, won’t be staged, and won’t be committed to the repository.

**Output**

****



# GIT-HOL

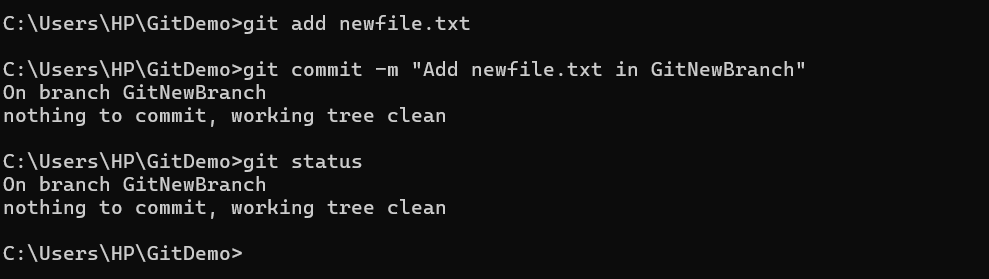
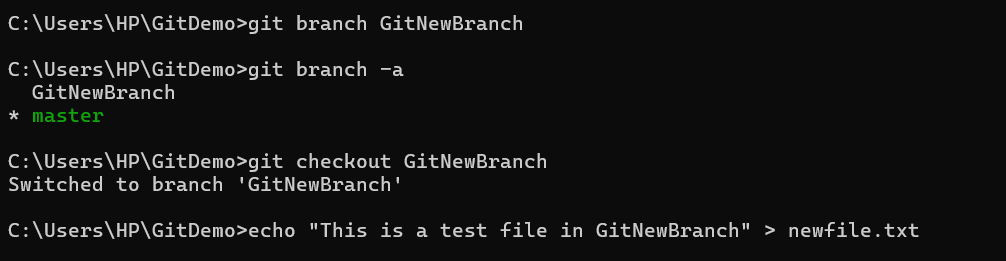
**Objectives**

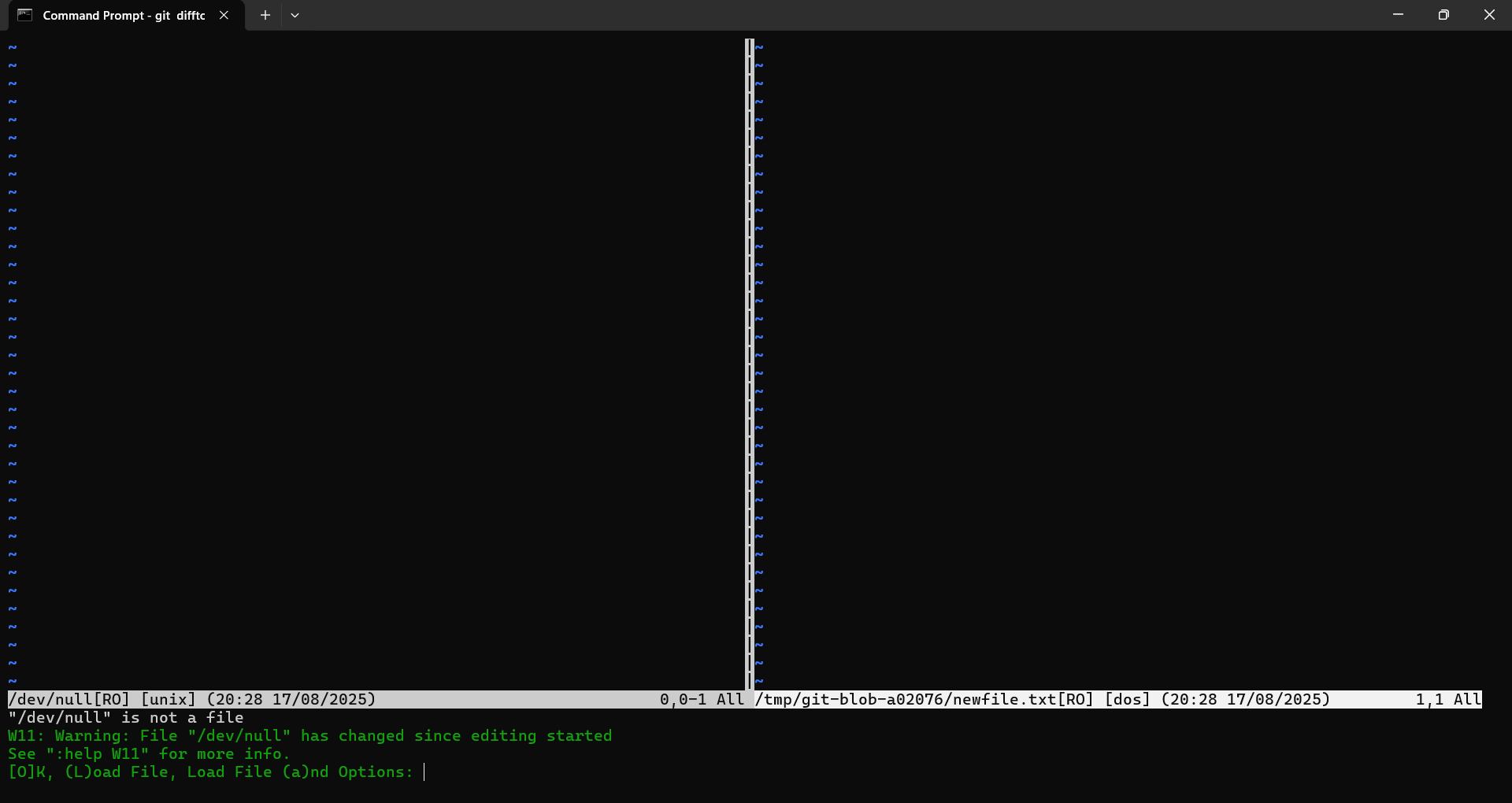
* + Explain branching and merging
  + Explain about creating a branch request in GitLab
  + Explain about creating a merge request in GitLab

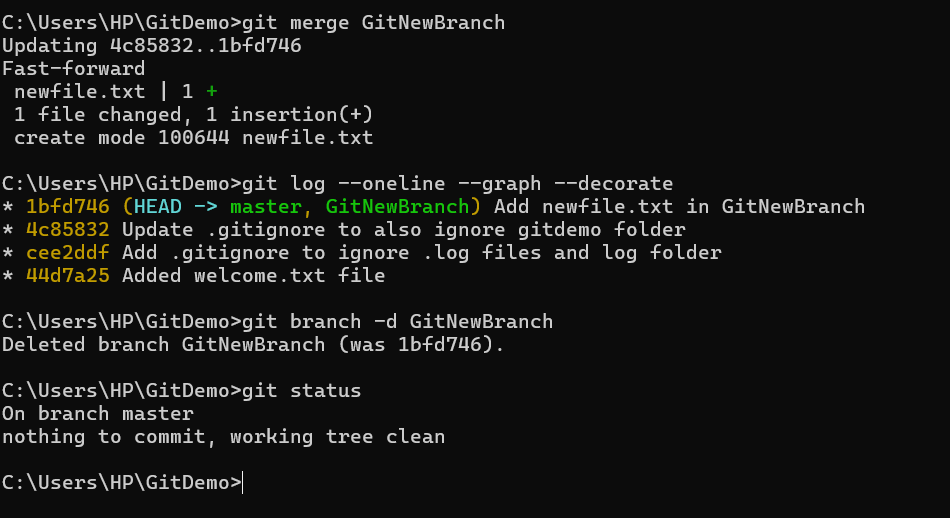
**Answers**

1. **Branching and Merging**
   * **Branching:** In Git, branching allows developers to create separate lines of development to work on new features, bug fixes, or experiments without affecting the main project.
   * **Merging:** Once work in a branch is complete, merging combines the changes back into another branch (usually main or master) to integrate the updates.
2. **Creating a Branch Request in GitLab**
   * A branch request is essentially creating a **new branch** in GitLab where you can push your changes before merging them into the main branch.
   * It helps in **isolating development work** and makes collaboration easier, as teammates can review and test changes independently.
3. **Creating a Merge Request in GitLab**
   * A merge request (MR) in GitLab is used to **propose merging changes** from one branch into another (e.g., feature-branch → main).
   * It enables **code review, discussion, and approval** before the changes are merged, ensuring better collaboration and quality control.

**Output**

****



****

# GIT-HOL

**Objectives:**

**How to resolve the conflict during merge**

1. **Identify and edit conflicted files:**

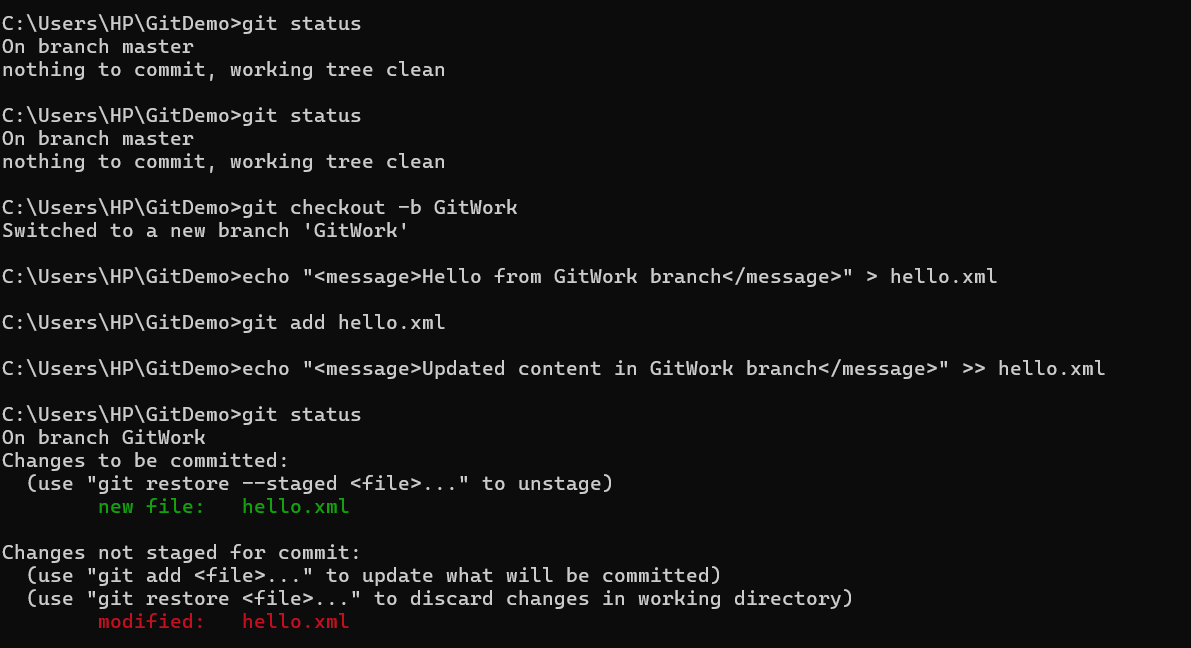
When a merge conflict occurs, Git marks the conflict inside the file with

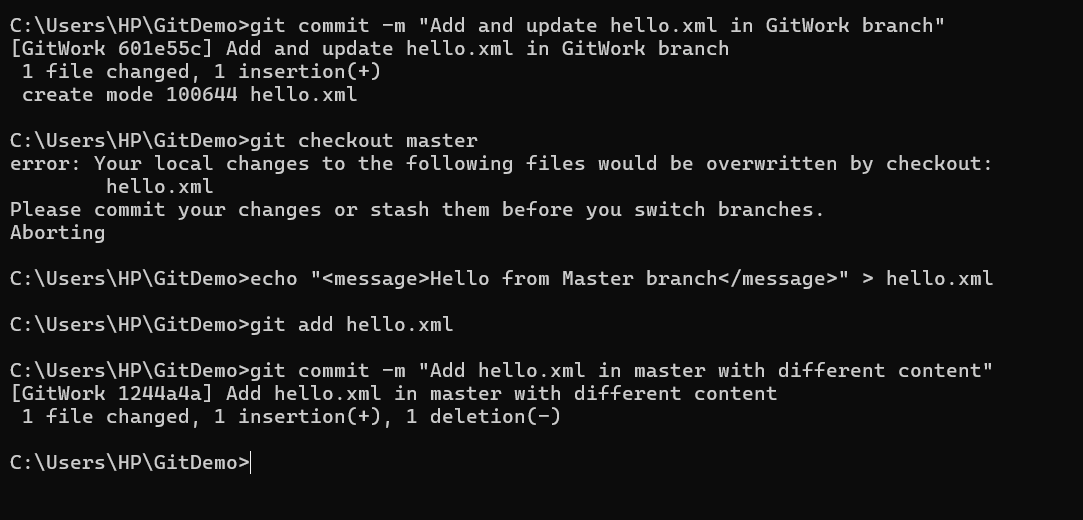
<<<<<<<, =======, and >>>>>>>. Open the file, review both versions, and manually edit it to keep the correct content.

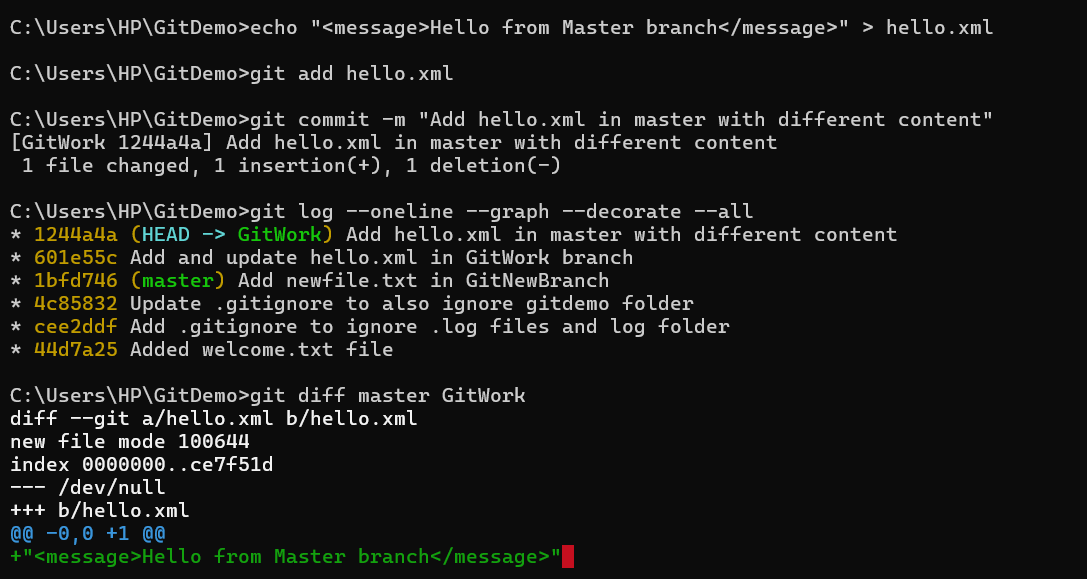
1. **Mark conflict as resolved and commit:**

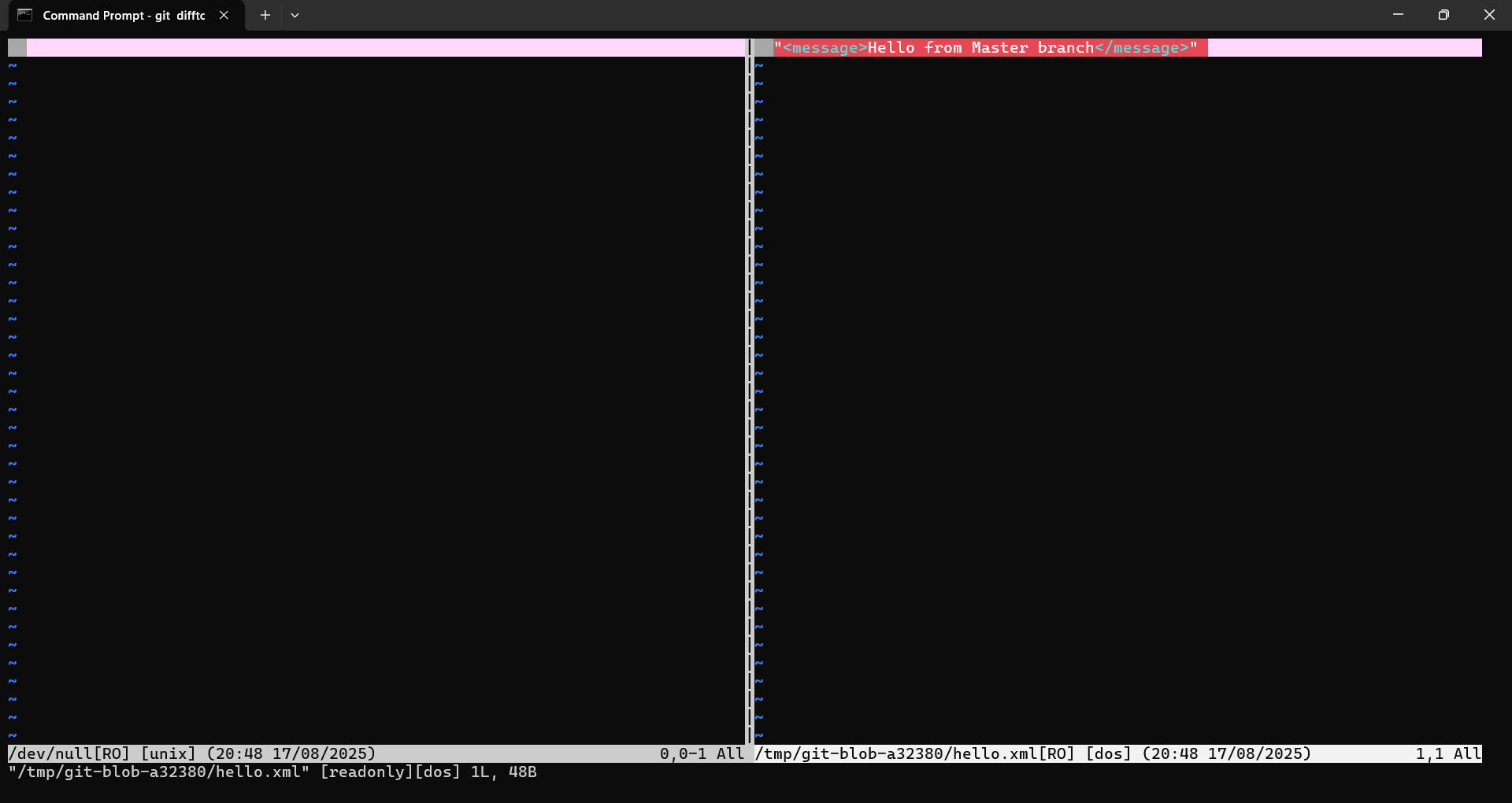
After editing, run git add <file> to mark the conflict as resolved, then complete the process with git commit to finalize the merge.

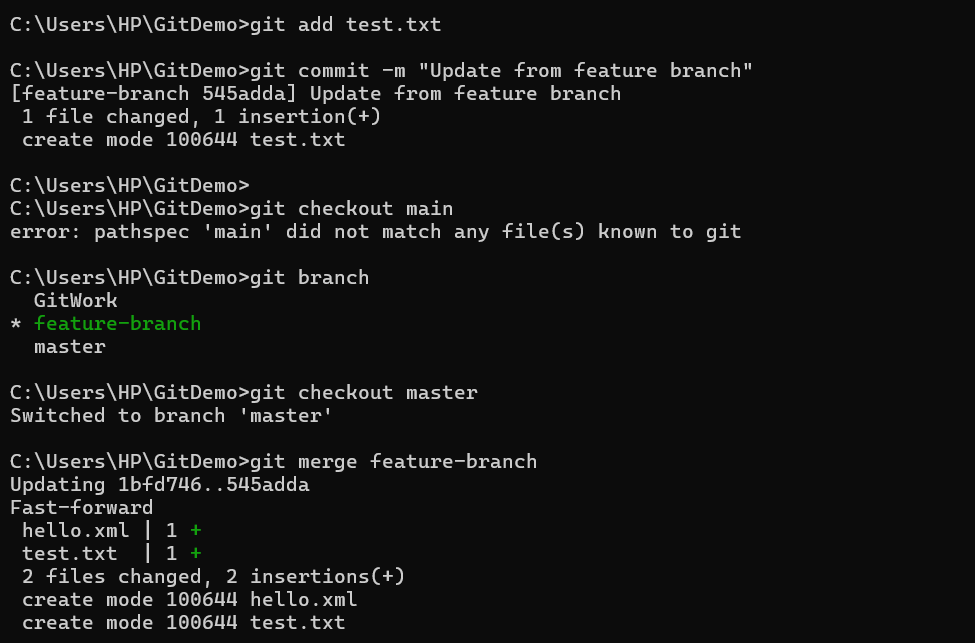
**Output**

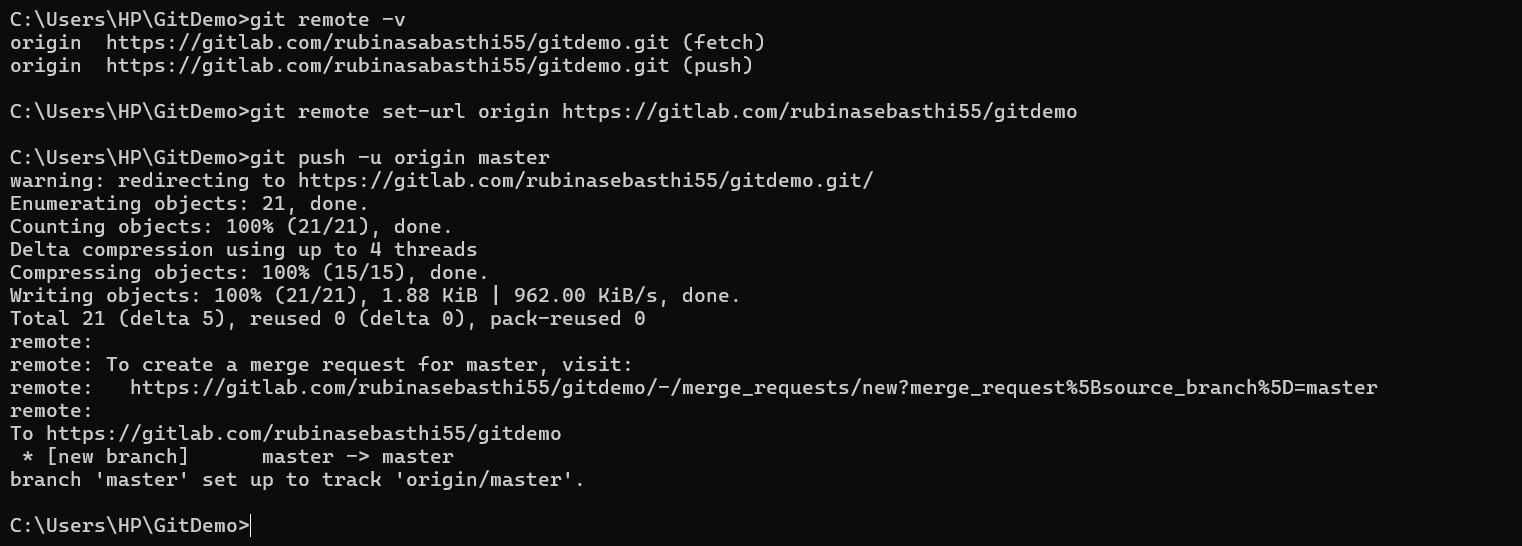
****

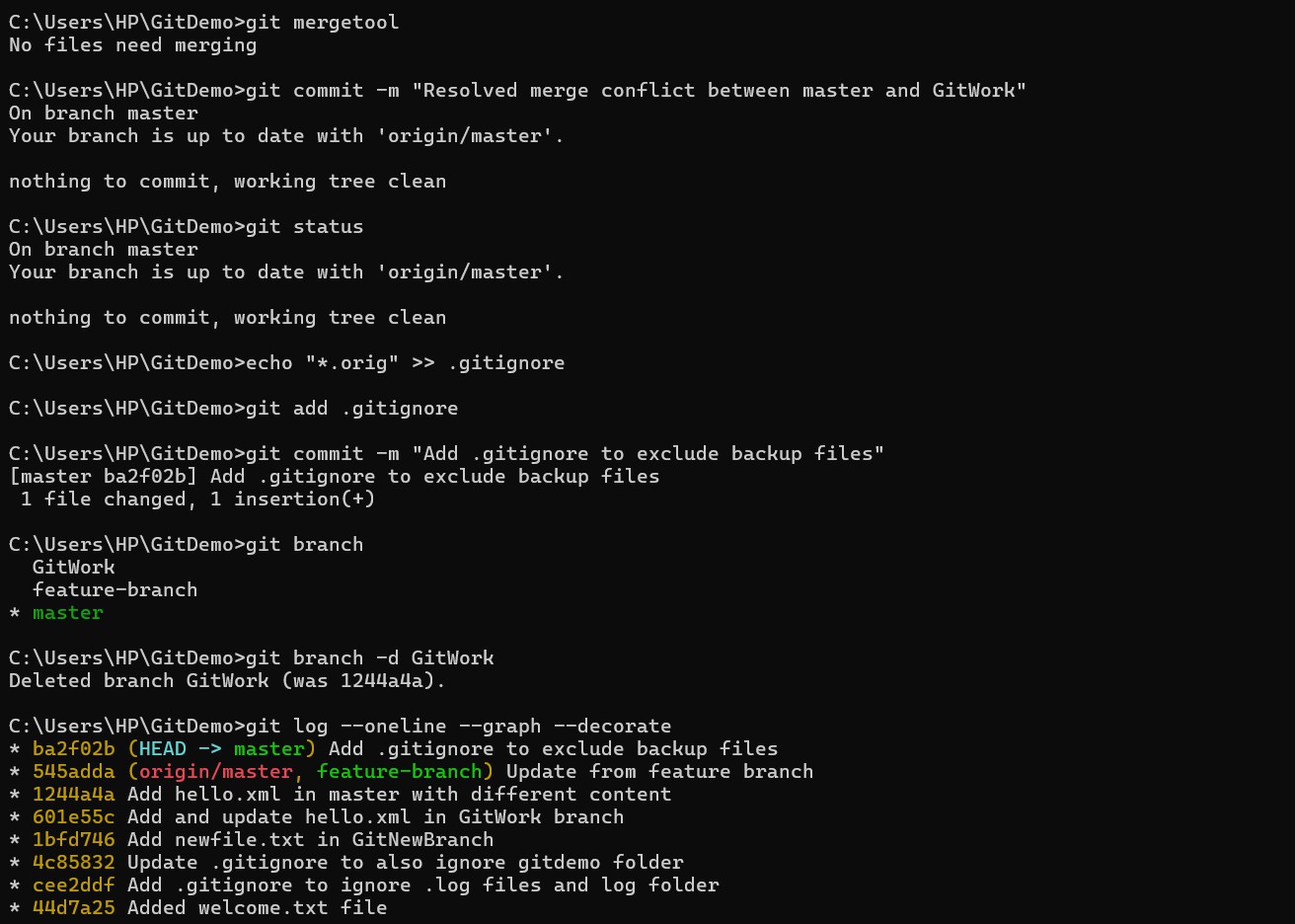


****



****



****

# GIT-HOL

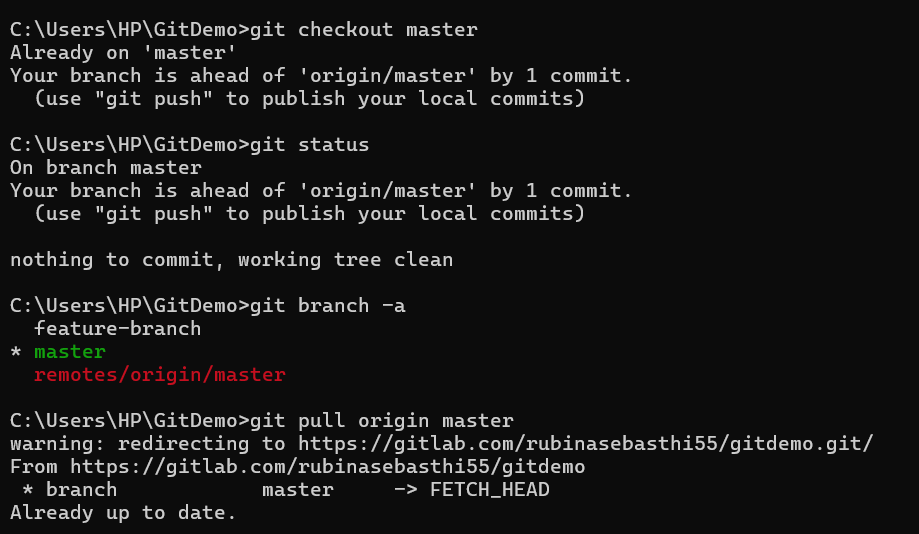
**Objectives**

1. Explain how to clean up and push back to remote Git

**Clean up local repository** – Remove unnecessary branches, commits, or files (using git branch -d, git reset, or git clean) to keep the local repo organized and error-free.

**Push changes to remote** – Use git push (or git push origin <branch>) to sync cleaned-up changes back to the remote repository, ensuring both local and remote are in sync.

**Output**

****

