**1. Git-HOL**

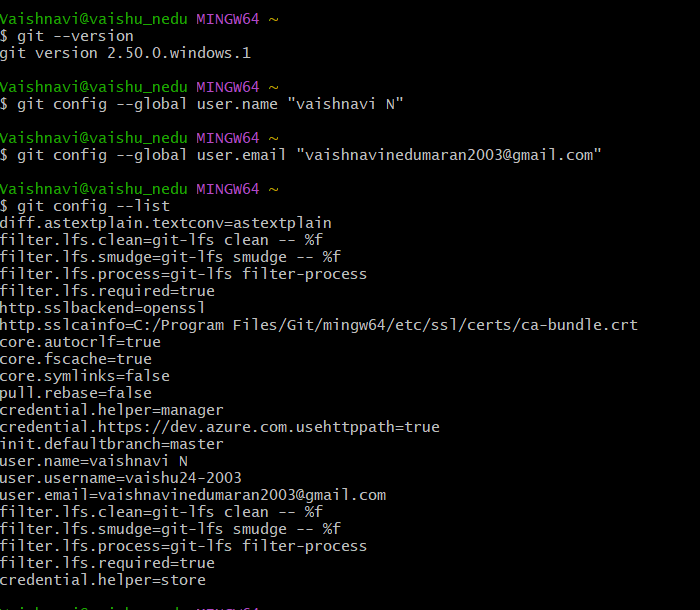
**Objectives**

Familiar with Git commands like git init, git status, git add, git commit, git push, and git pull.

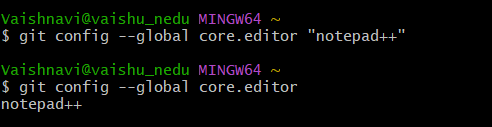
In this hands-on lab, you will learn how to

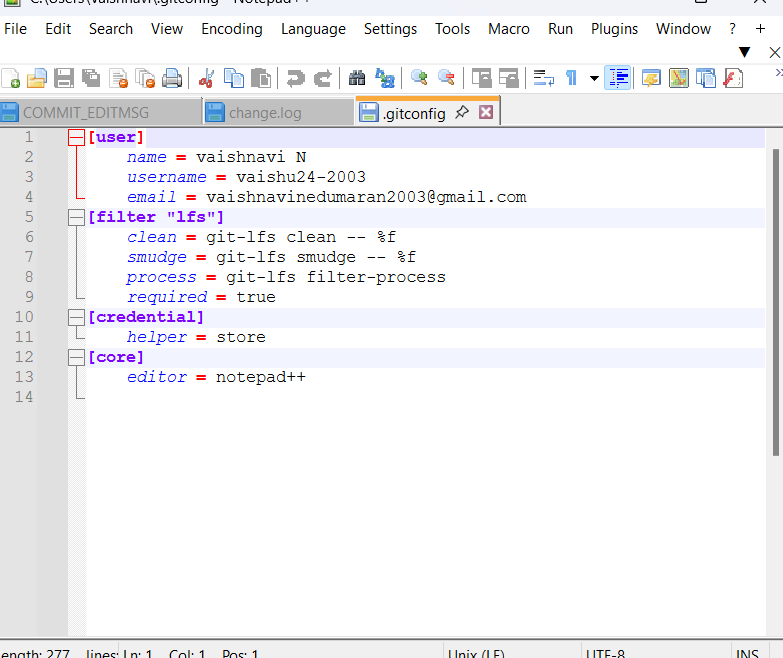
* Setup your machine with Git Configuration
* Integrate notepad++.exe to Git and make it a default editor
* Add a file to source code repository

**Step 1: Git Configuration**

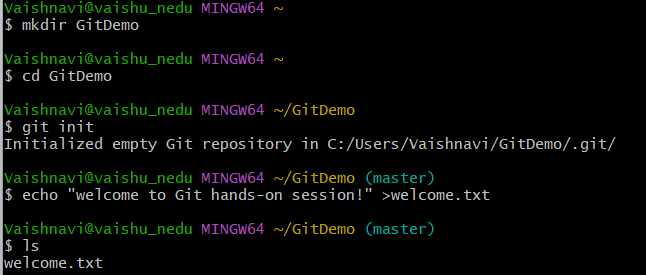


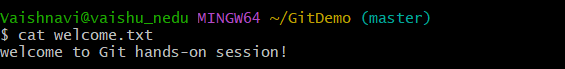
**Step 2: Integrate notepad++.exe to Git and make it a default editor**

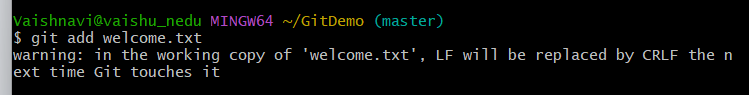


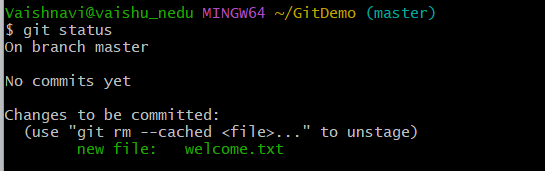


**Step 3: Add a file to source code repository**

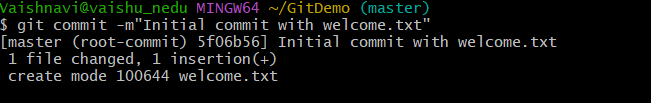


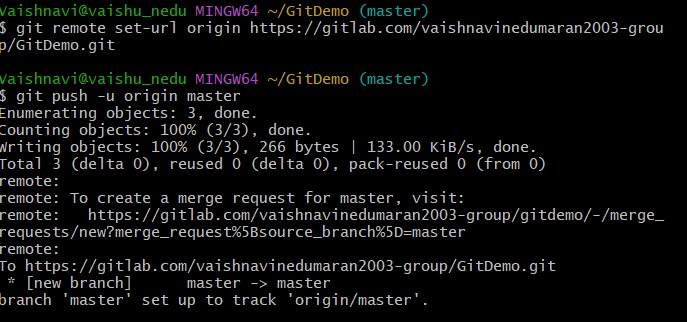






**Step 4: Signup with GitLab and create a remote repository “GitDemo”**





**2. Git-HOL**

**Objectives**

**🡪 Explain git ignore**

.gitignore is a special file in your Git repository that tells Git:

"Do **not track** these files or folders."

It's used to ignore:

* Temporary files (like .log, .tmp)
* Build/output folders (like node\_modules/, target/, log/)
* IDE files (.vscode/, .class files, etc.)

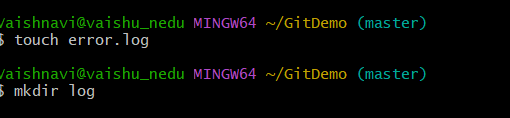
In this hands-on lab, you will learn how to:

🡪Implement git ignore command to ignore unwanted files and folders

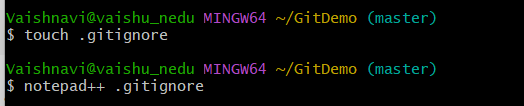
**🡪Explain how to ignore unwanted files using git ignore**

**Git Ignore Implementation:**

1. **Created unwanted file and folder:**

****

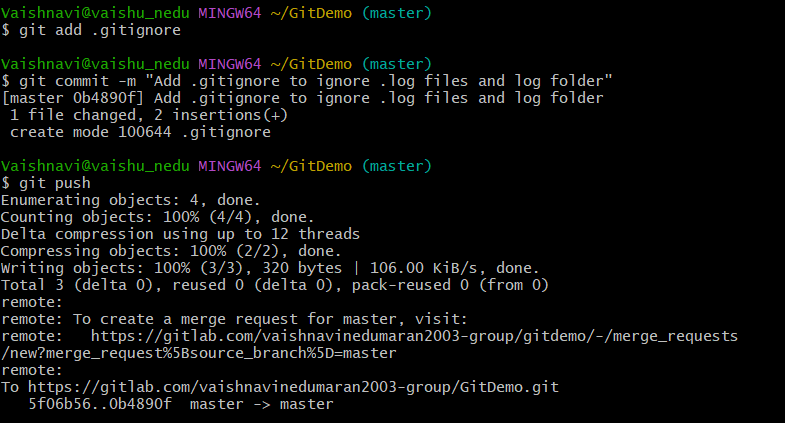
**2.Created .gitignore file and added:**

****

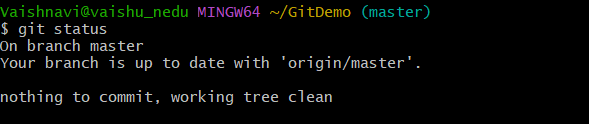
**\*.log**

**log/ (added inside the notepad++)**

**3.Added and committed .gitignore:**

****

**4.** **Verified using:**

****

**3.Git-HOL**

**Objectives**

Explain branching and merging

Explain about creating a branch request in GitLab

Explain about creating a merge request in GitLab

In this hands-on lab, you will learn how to:

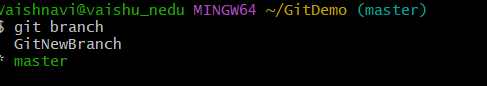
Construct a branch, do some changes in the branch, and merge it with master (or trunk)

**Part 1: Branching**

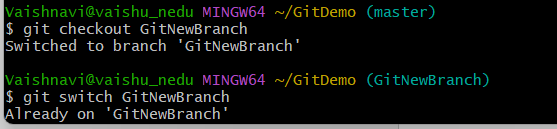
**Step 1: Create a new branch**

****

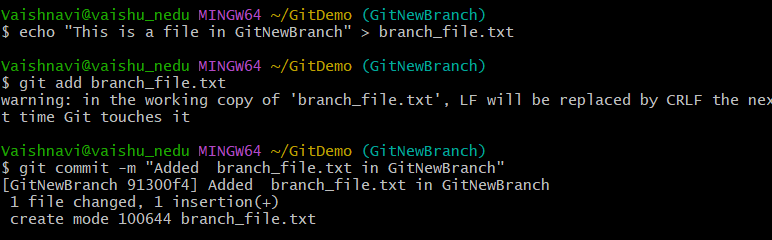
**Step 2: List all local branches**

****

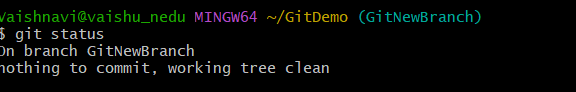
**Step 3: Switch to the new branch**

****

**Step 4: Create a new file in the branch**

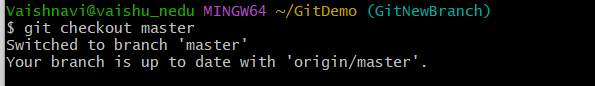
****

**Step 5: Check branch status**

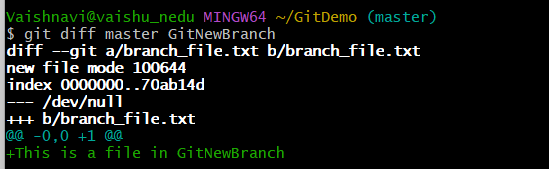
****

**Part 2: Merging**

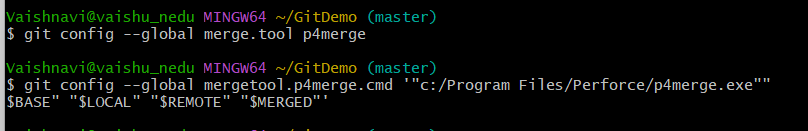
**Step 1: Switch back to master**

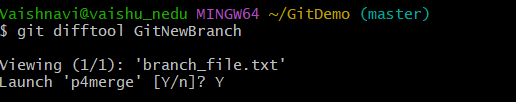
****

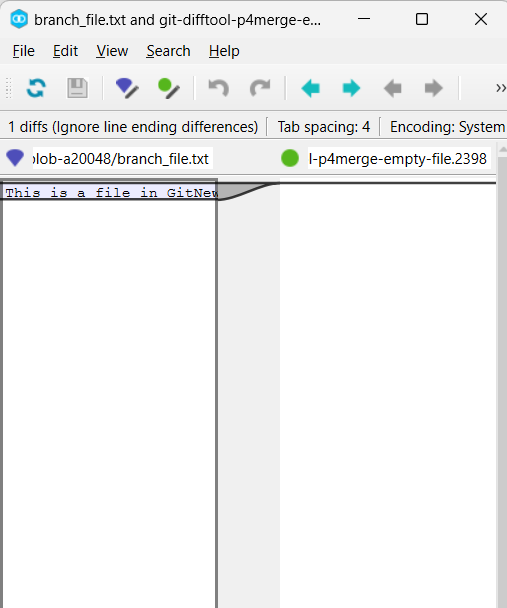
**Step 2: See differences between branches (CLI)**

****

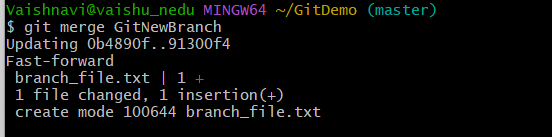
**Step 3: See visual differences (P4Merge)**

****

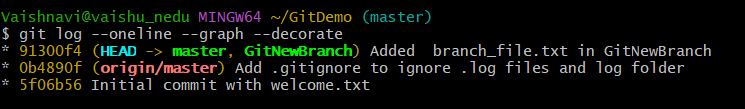
****

****

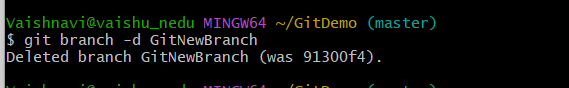
**Step 4: Merge GitNewBranch into master**

****

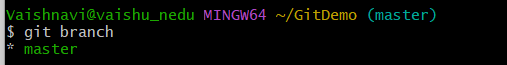
**Step 5: View merge history visually**

****

**Step 6: Delete the merged branch**

****

**Verify:**

****

**🡪** **Successfully practiced Git branching and merging using CLI and visual tools.**

**4. Git-HOL**

**Objectives**

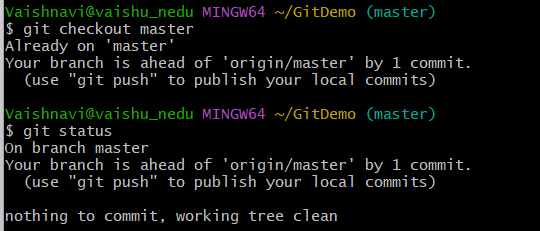
Explain how to resolve the conflict during merge.

In this hands-on lab, you will learn how to:

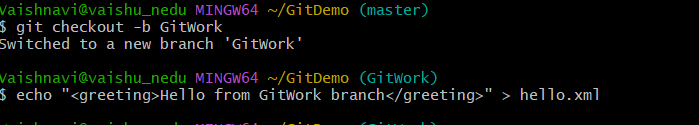
* Implement conflict resolution when multiple users are updating the trunk (or master) in such a way that it results into a conflict with the branch’s modification.

**Step-by-Step: Conflict Resolution Hands-On Lab**

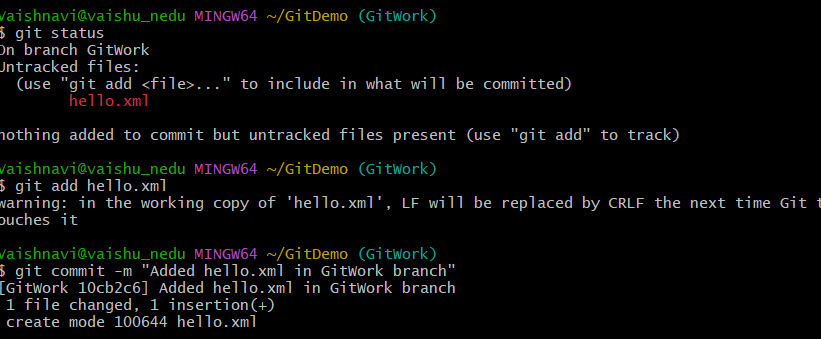
**1.Verify master is in a clean state**

****

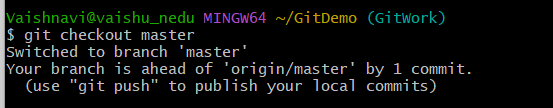
**2.Create branch GitWork and add hello.xml**

****

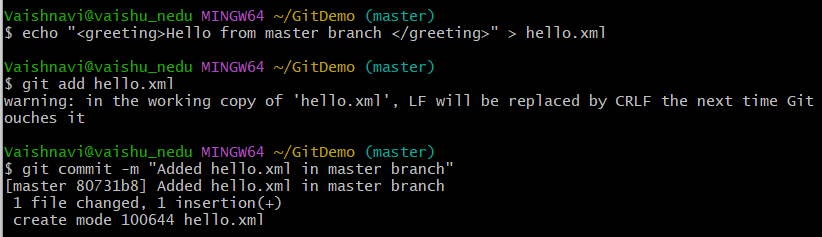
**3. Check status and commit**

****

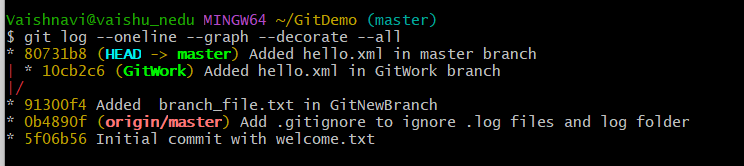
**4.** **Switch back to master**

****

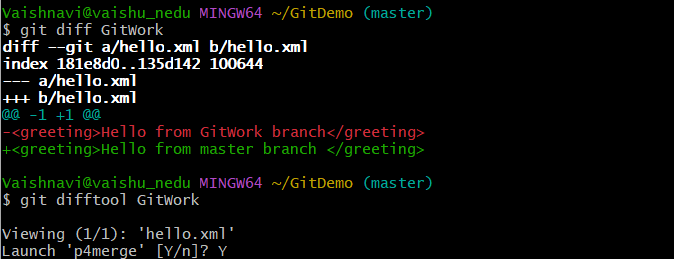
**5.** **Create same hello.xml with different content**

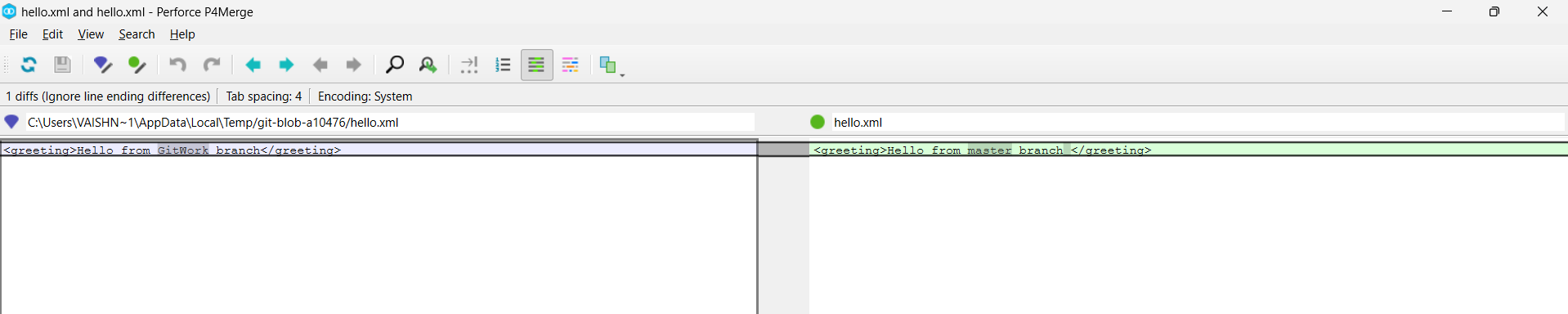
****

**6.** **Observe log (visualize both branches)**

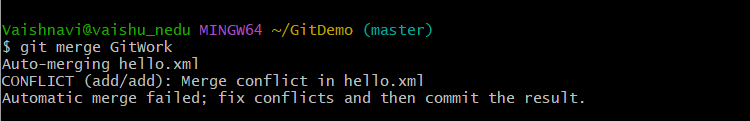
****

**7.** **Check differences (CLI and GUI)**

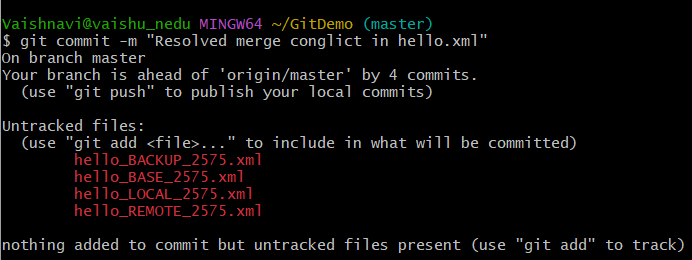
****

****

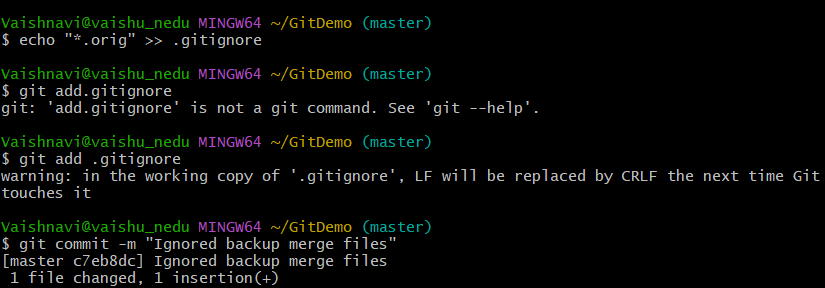
**8. Merge the branch and trigger conflict**

****

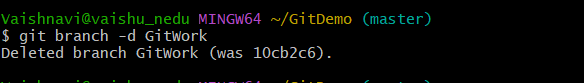
**9. Use 3-way merge with P4Merge**

****

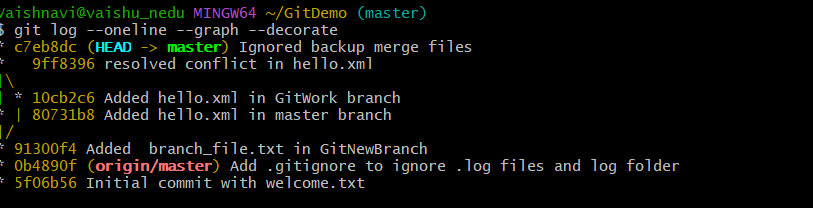
**10.** **Add backup file to .gitignore**

****

**11.** **Clean up merged branch**

****

**12. Final log view**

****

**5.Git-HOL**

**Objectives**

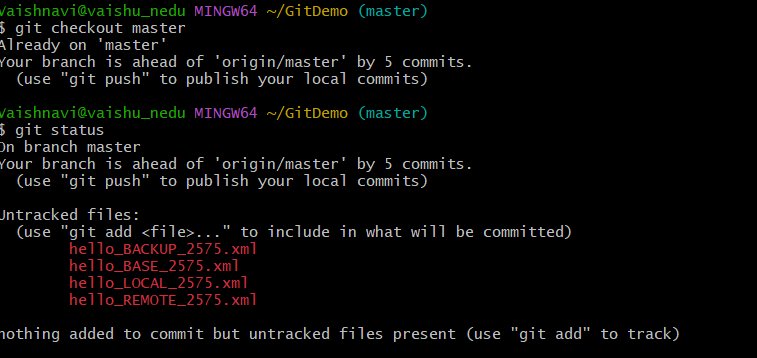
Explain how to clean up and push back to remote Git

In this hands-on lab, you will learn how to:

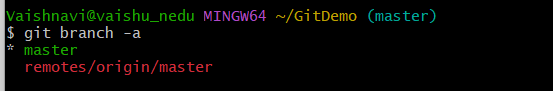
🡪Execute steps involving clean up and push back to remote Git.

**Step-by-Step Instructions**

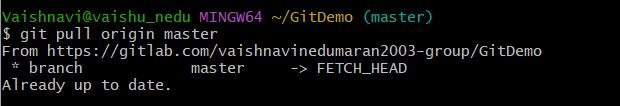
1. **Verify if master is in clean state**

****

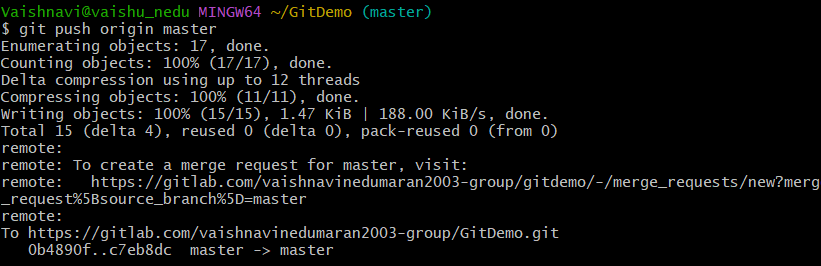
1. **List all available branches**

****

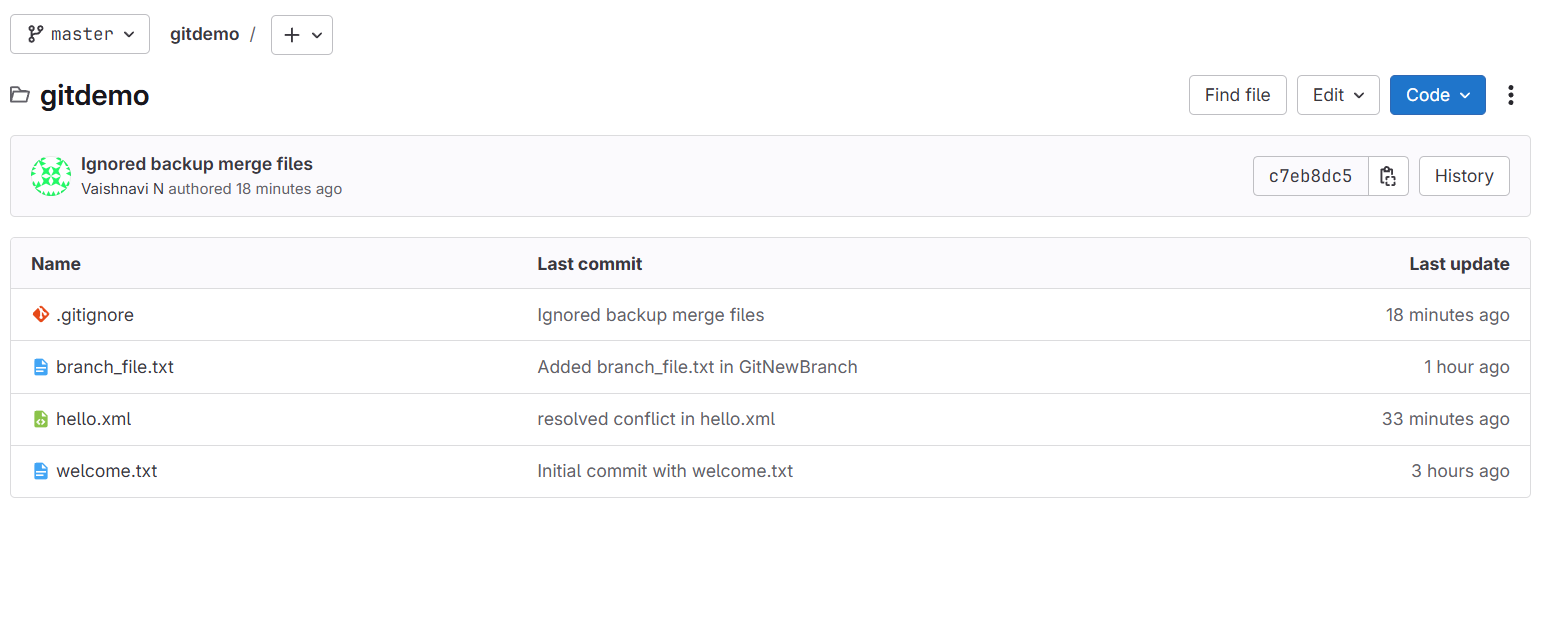
1. **Pull remote repository**

****

1. **Push all local changes to remote**

****

1. **Verify changes on GitLab**

****