**DN 4.0 JAVA FSE SOLUTIONS – WEEK 8**

**SKILL: GIT**

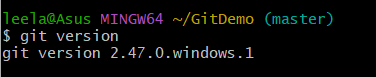
**HandsOn 1:**

**Objectives -**

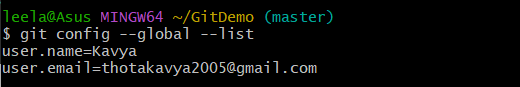
Familiar with Git commands like git init, git status, git add, git commit, git push, and git pull.

**Step 1: Setup your machine with Git Configuration**

1. Knowing git version -



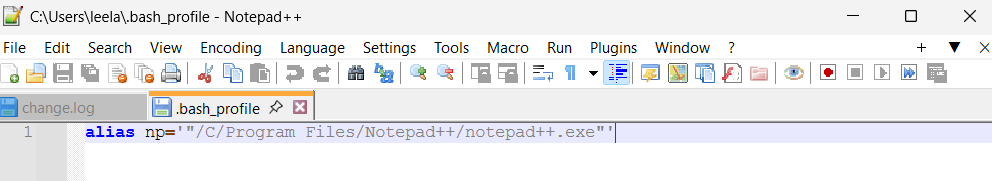
1. Checking configuration -



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

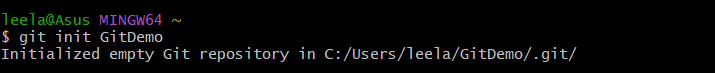
Notepad++ is integrated and made it a default editor



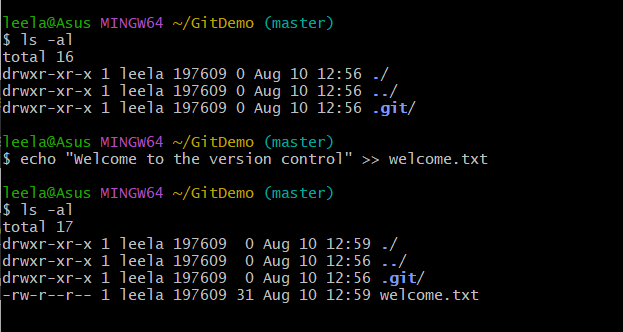


**Step 3: Add a file to source code repository**

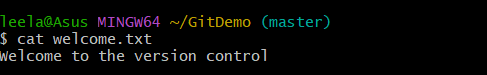
1. Initialzing -



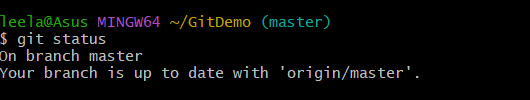
1. Cmd: ls -al



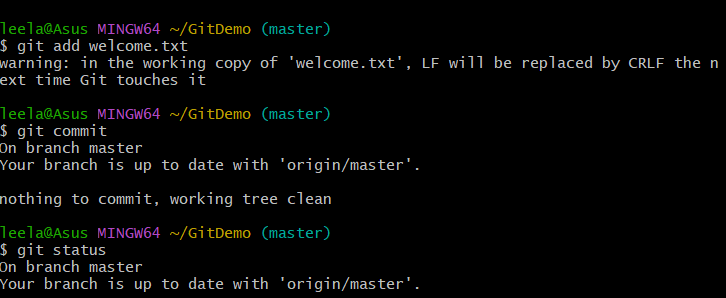
1. Verifying content -



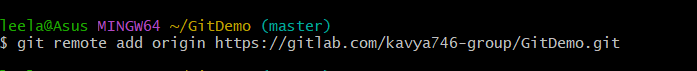
1. Checking git status -

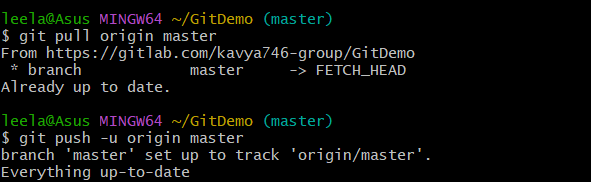


1. Adding welcome.txt and commit -

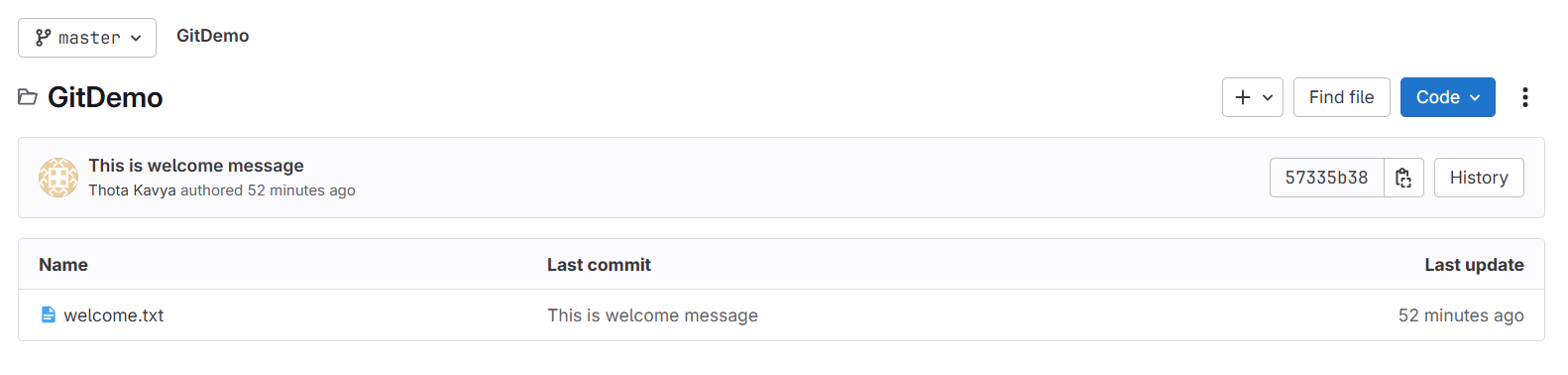


1. Signed in with gitlab and created GitDemo repository -





**Output:**

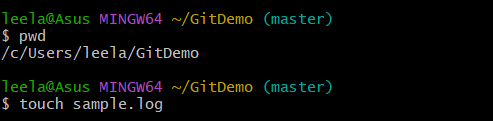
****

**HandsOn 2:**

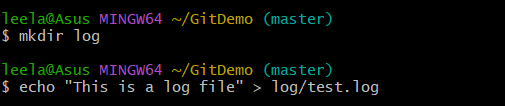
**Objectives -**

Create a “.log” file and a log folder in the working directory of Git. Update the .gitignore file in such a way that on committing, these files (.log extensions and log folders) are ignored.

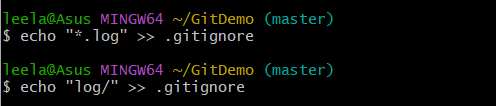
**Step-1:** Creating a log file



**Step-2:** Creating a log folder

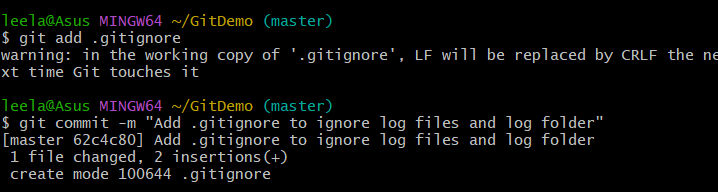


**Step-3:** Creating .gitignore and adding rules

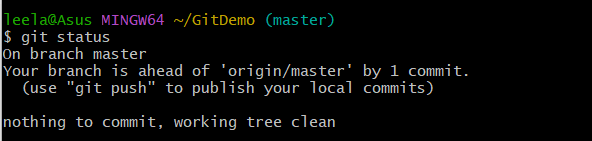


**Step-4: Local Repository Status -** Only .gitignore is tracked.

**Stage and commit .gitignore -**

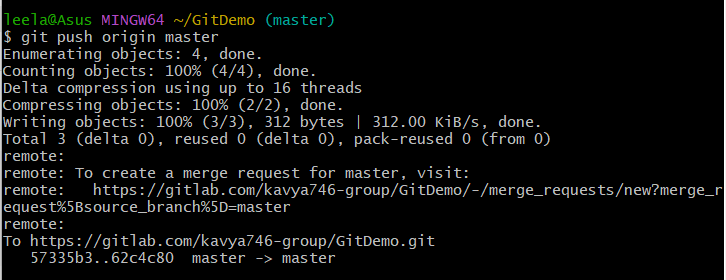


**Step-5:** Verify with git status

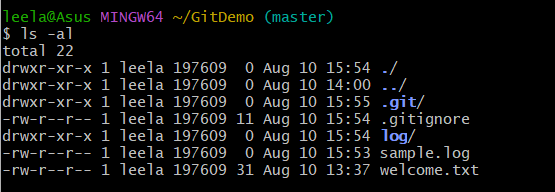


We can’t see sample.log or the log/folder listed as untracked because they’re ignored.

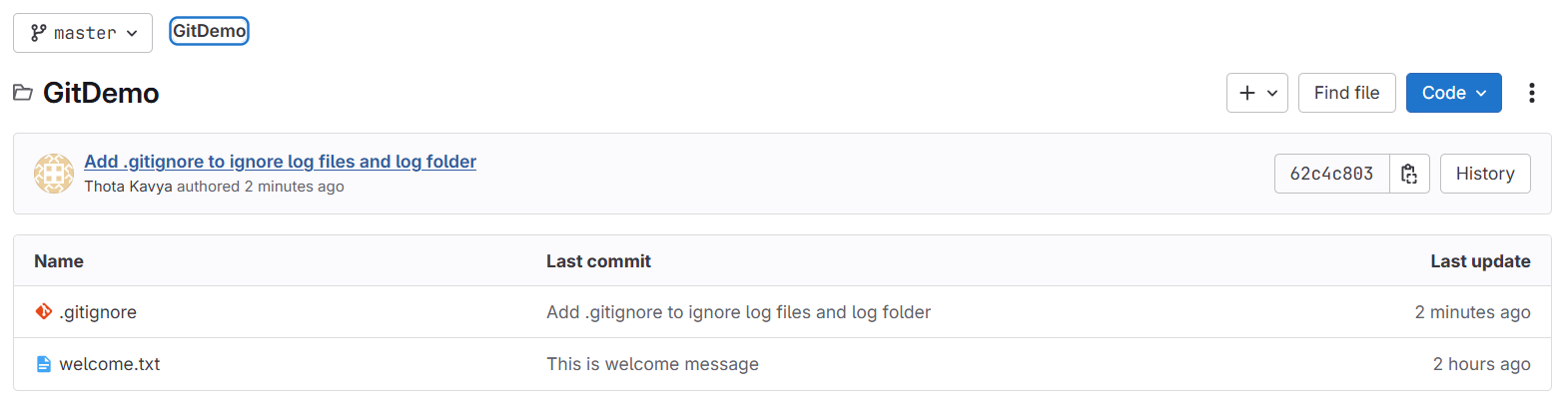
**Step-6:** Push to remote



**Working Directory Status:**

****

**Git Repository Status:**

****

**HandsOn 3:**

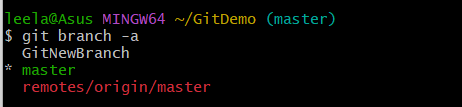
**Objective - Branching and Merging**

**Branching:**

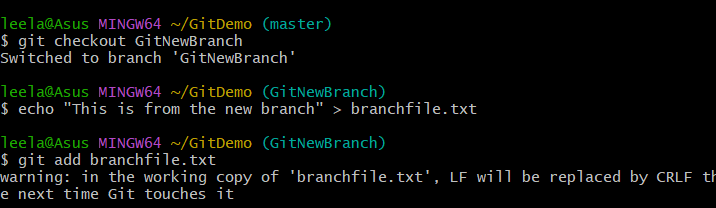
1. Create a new branch “GitNewBranch”



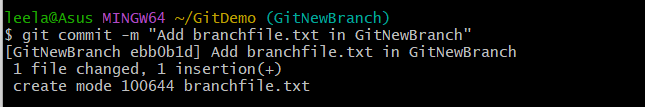
1. List all the local and remote branches



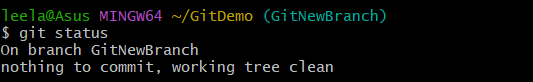
1. Switch to the newly created branch and add files



1. Commit the changes

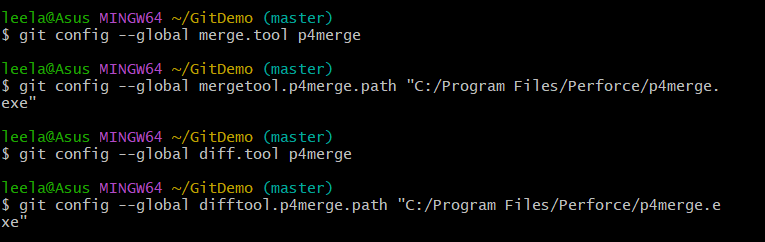


1. Check the status

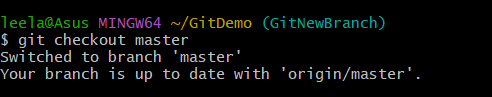


**Merging:**

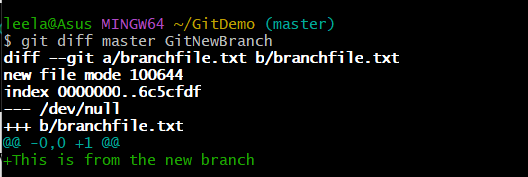
**Commands used for Git to use P4Merge -**

****

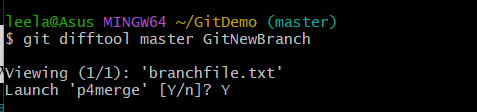
1. Switch to the master

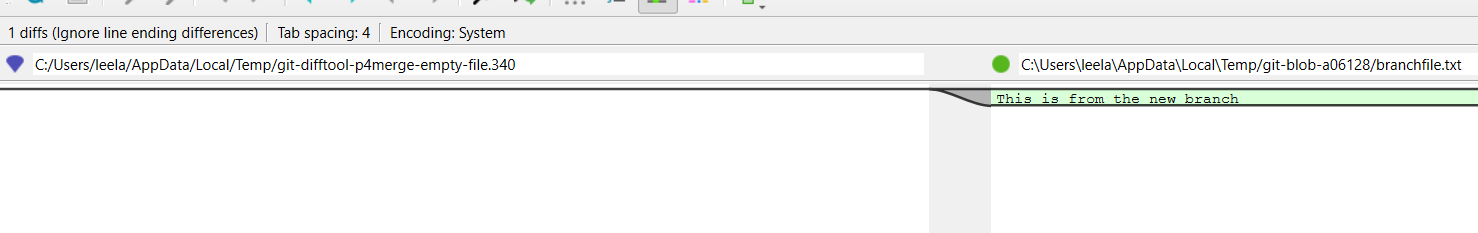


1. List out all the differences between trunk and branch

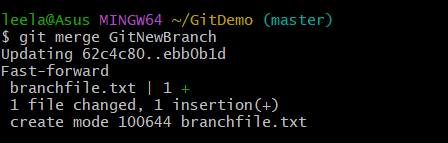


1. List out all the visual differences between master and branch using **P4Merge tool**

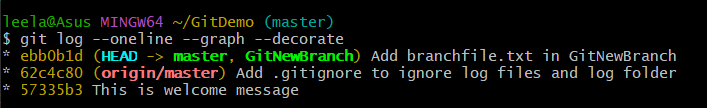


****

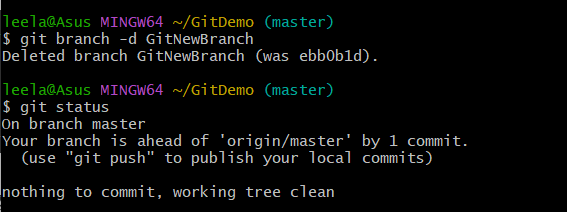
1. Merge the source branch to the trunk

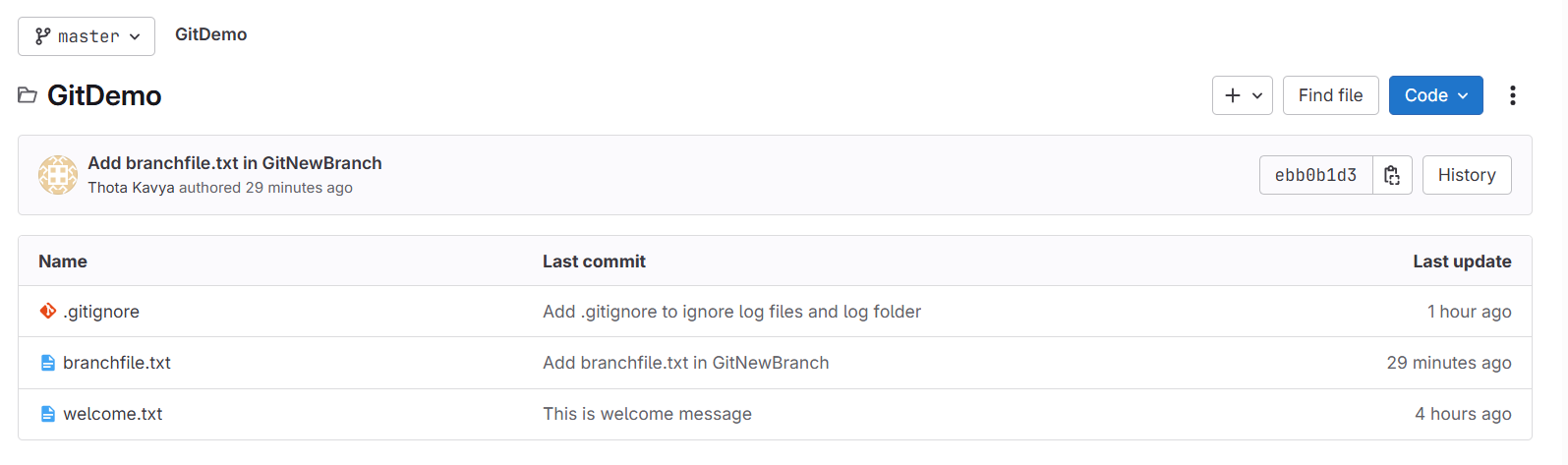


1. Observe the logging after merging



1. Delete the branch after merging and observe status

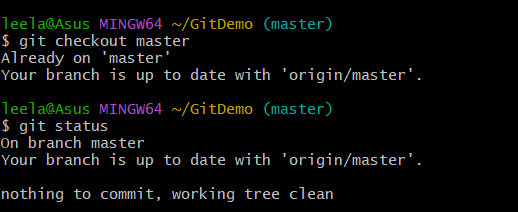




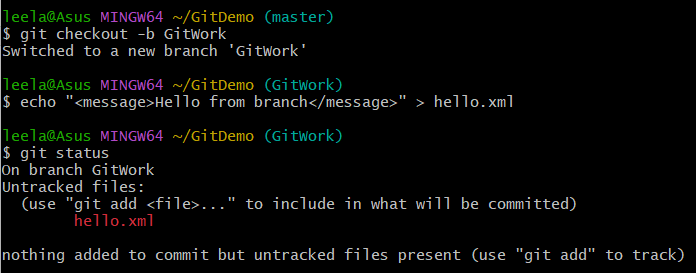
**HandsOn 4:**

**Objective - Explain how to resolve the conflict during merge**

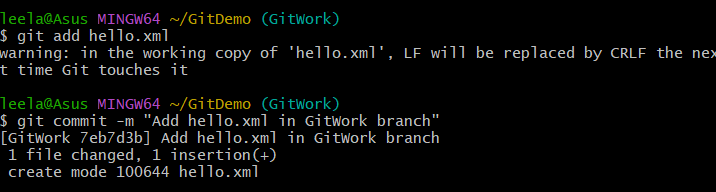
1. Verify master is clean



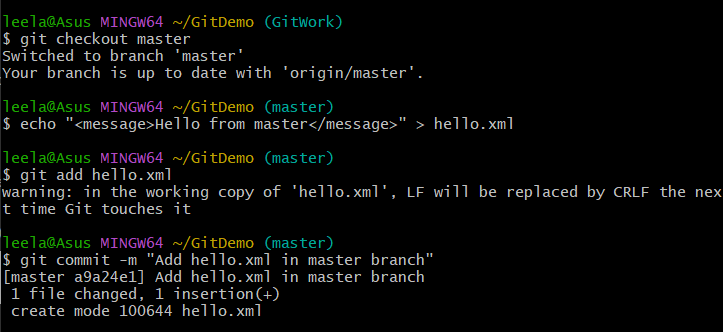
1. Create branch and add file



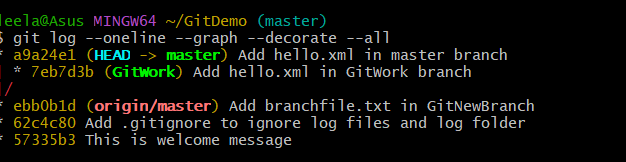
1. Commit changes in branch



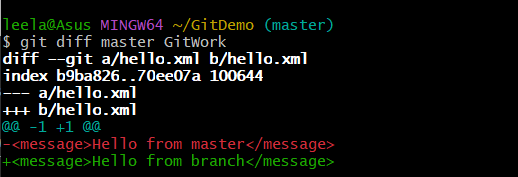
1. Switch to master and create conflicting file



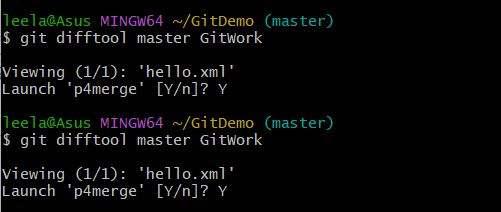
1. View logs

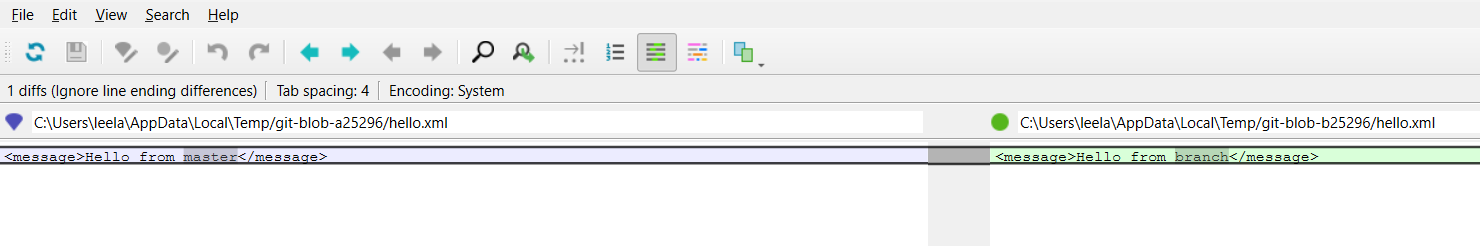


1. Check differences in command-line

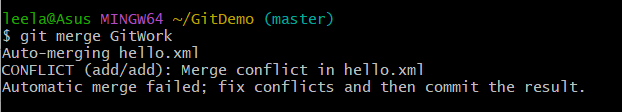


1. View differences in P4Merge

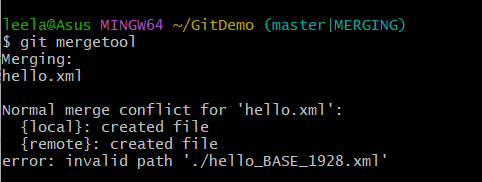


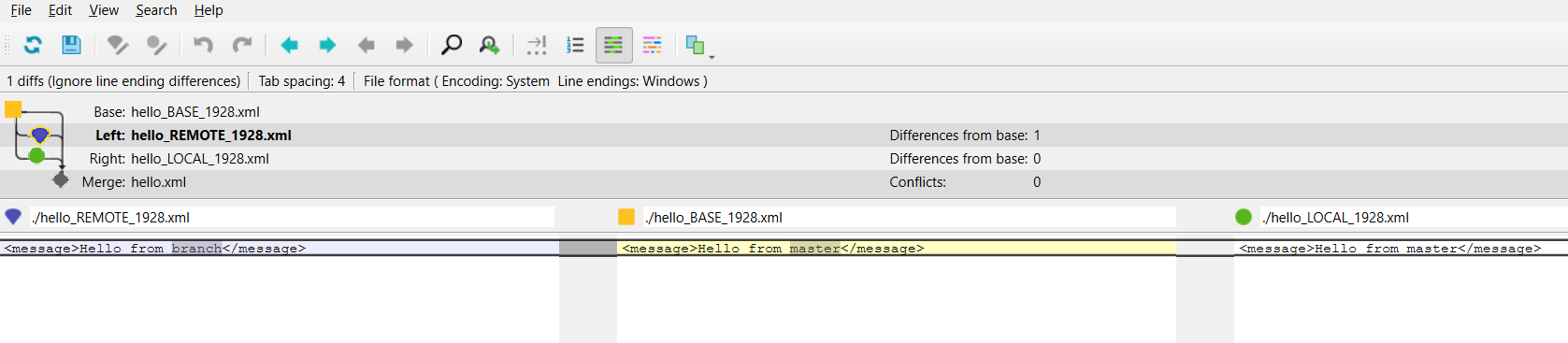


1. Attempt merge - cause conflict

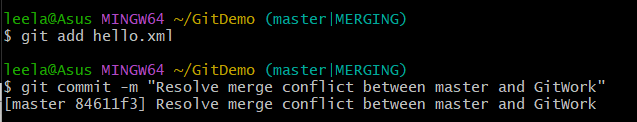


1. Resolve conflict with 3-way merge in P4Merge

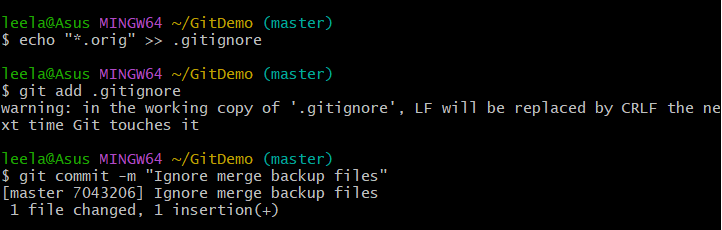




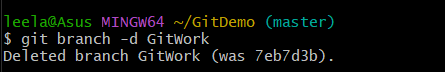
10)Mark conflict as resolved and commit



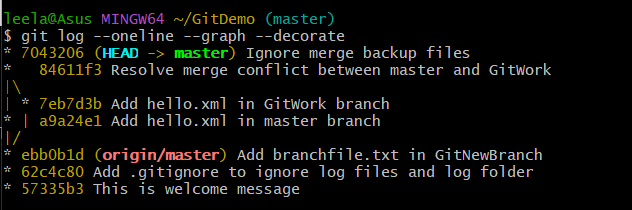
11)Add backup file from merge to .gitignore

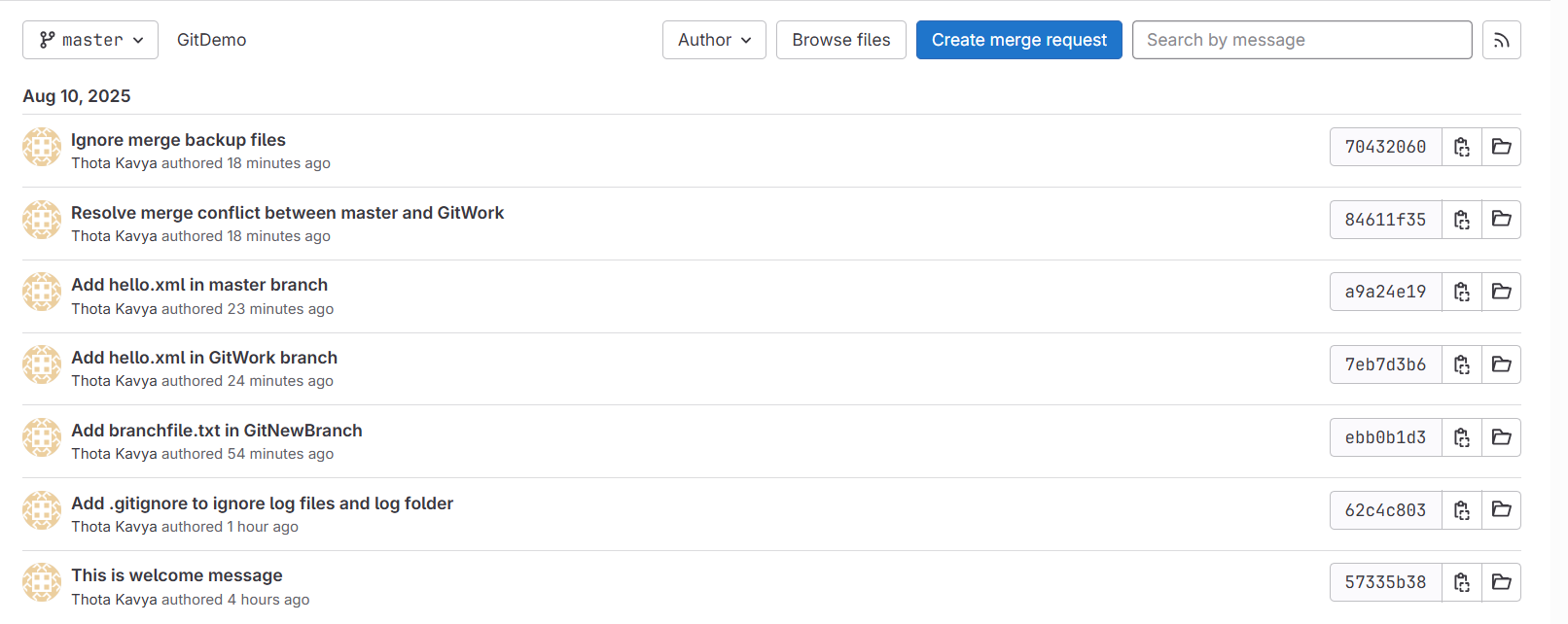


12) Cleanup branch



13) Final log view

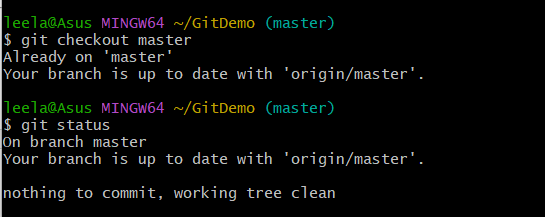




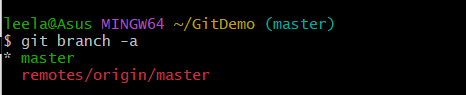
**HandsOn 5:**

**Objective - Explain how to clean up and push back to remote Git**

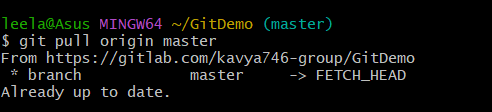
1. Verify master is clean



1. List all branches



1. Pull latest changes from remote



1. Push pending changes from Git-T03-HOL\_002

