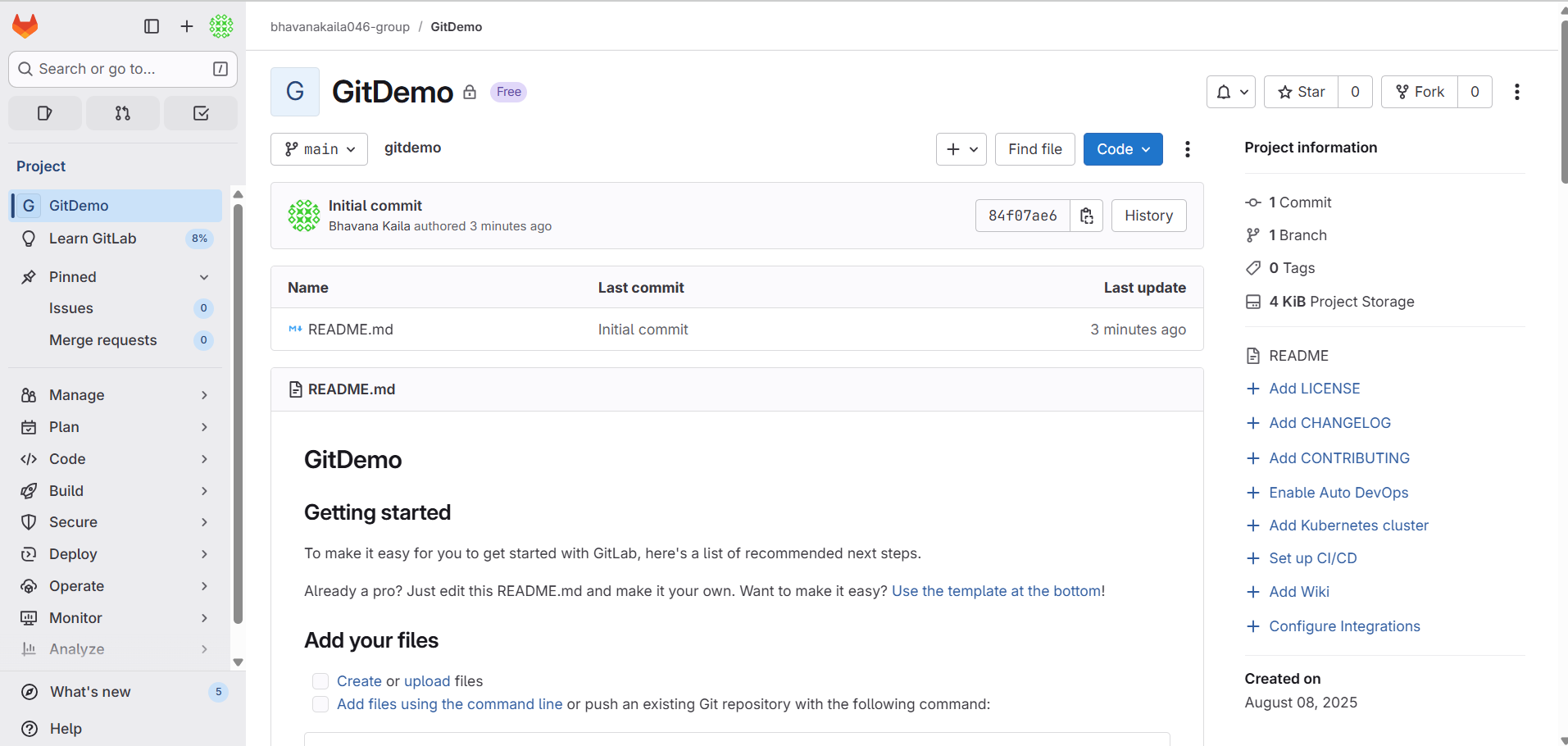
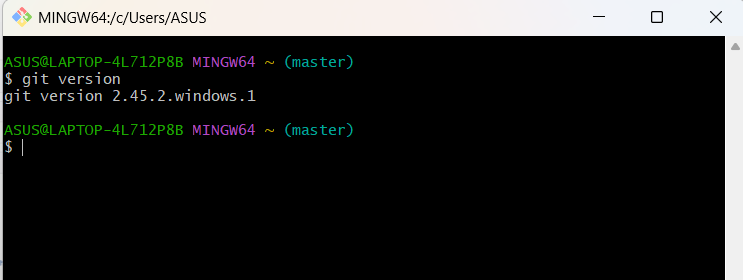
**1.Git-HOL:**

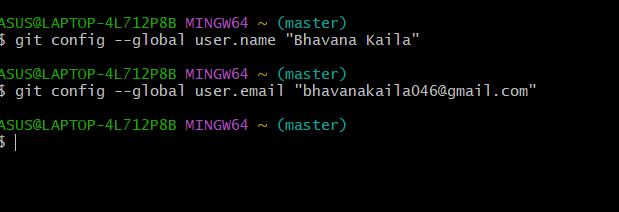
**Step 1: Setup your machine with Git Configuration**

1. Signup with GitLab and register your credentials. Login to GitLab and create a “GitDemo” project.

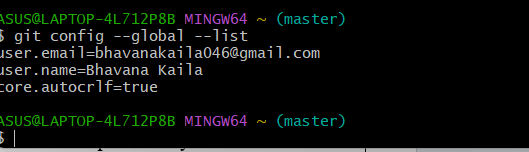


2. To check if Git client is installed properly: Open Git bash shell and execute



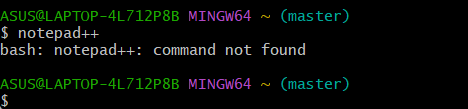
3. To configure user level configuration of user ID and email ID execute(use GitLab credentials)

4. To check if the configuration is properly set, execute the following command



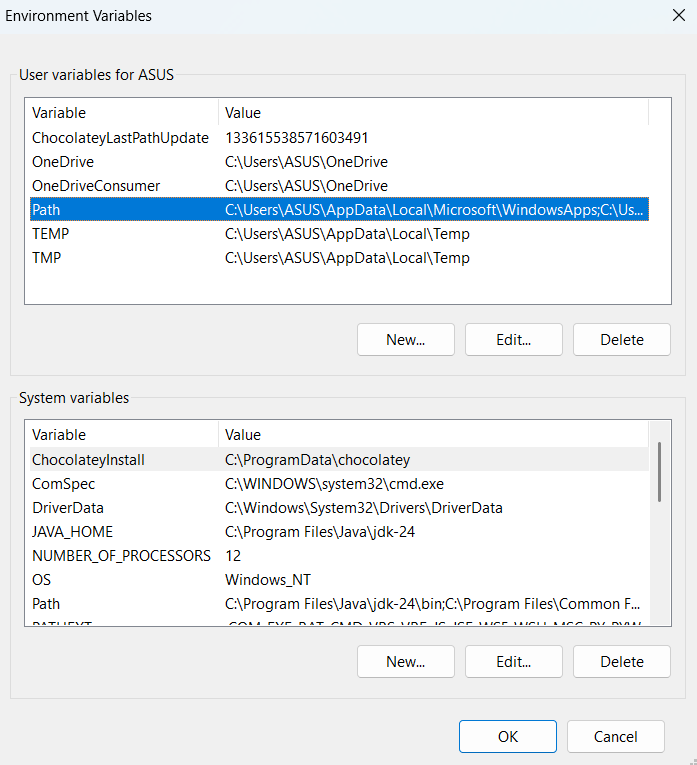
**Step 2: Integrate notepad++.exe to Git and make it a default editor**

1.To check, if notepad++.exe execute from Git bash

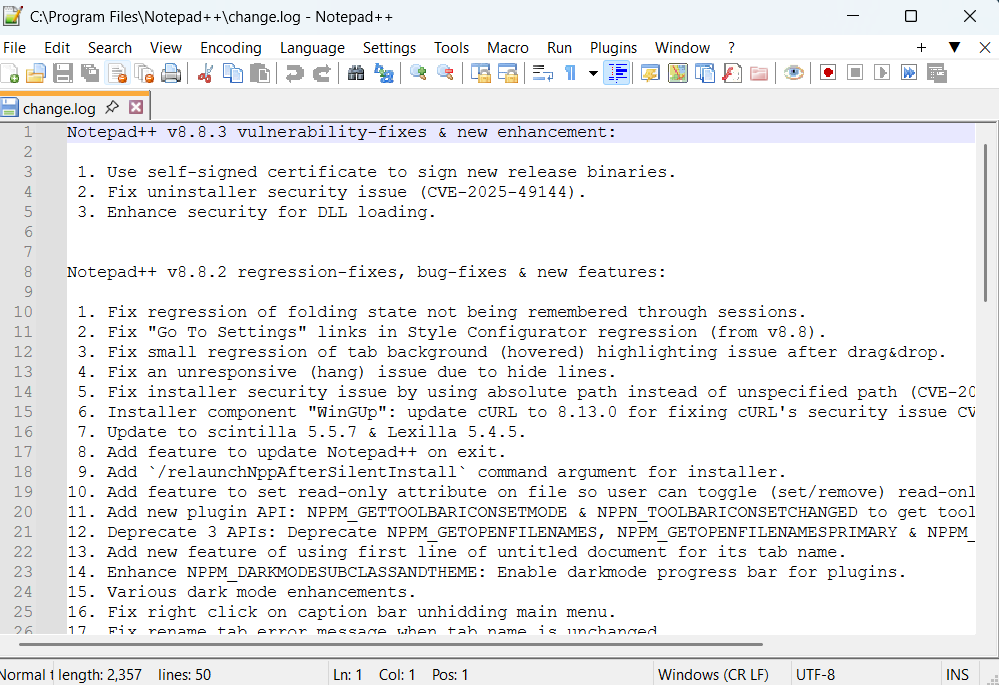
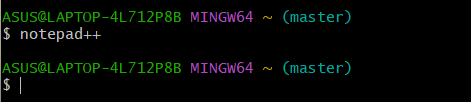


2. If Git bash could not able to recognize notepad++ command that implies notepad++.exe is note added to the environment path variable.

To add path of notepad++.exe to environment variable, go to control panel -> System -> Advanced System settings. Go to Advanced tab -> Environment variables -> Add path of notepad++.exe to the path user variable by clicking on “Edit”

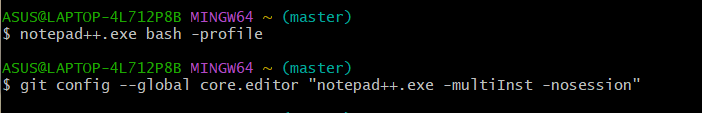


3. Exit Git bash shell, open bash shell and execute “notepad++.exe”, Now, notepad++ will open from Git bash shell

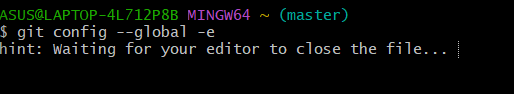


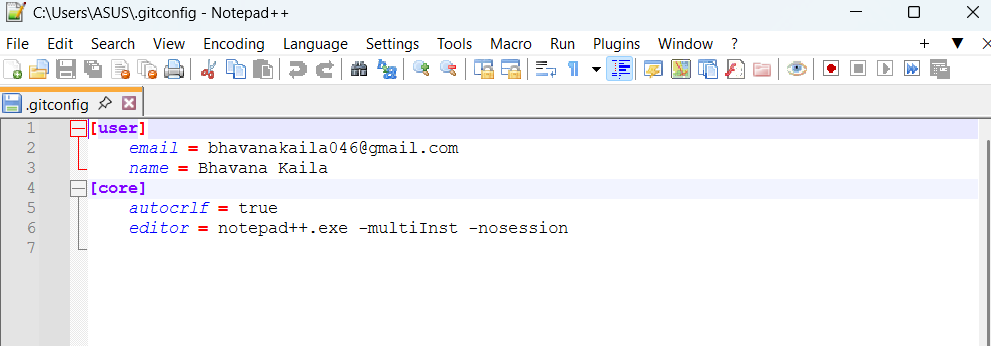
4. To create an alias command for notepad++.exe, execute “notepad++.exe bash -profile”, It will open notepad++ from bash shell, and create a user profile by adding the line in notepad++.

5. To configure the editor, execute the command



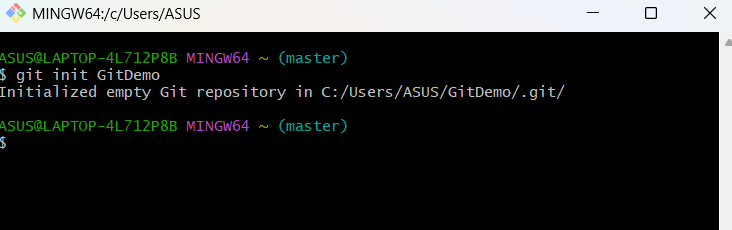
6. To verify if notepad++ is the default editor, execute the command



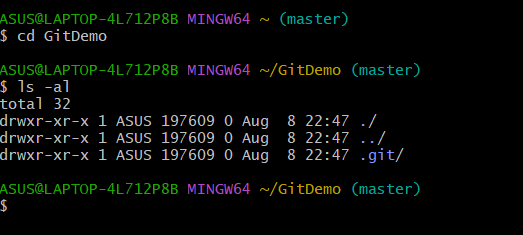


**Step 3: Add a file to source code repository**

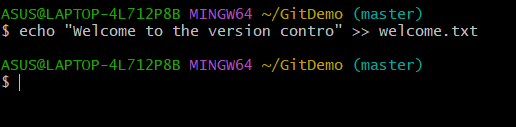
1.Open Git bash shell and create a new project “**GitDemo**” by executing the command



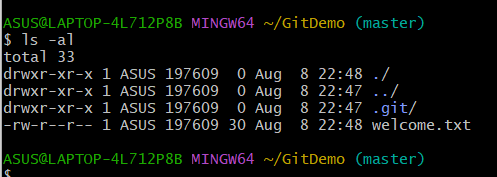
2. Git bash initializes the “**GitDemo**” repository. To verify, execute the command



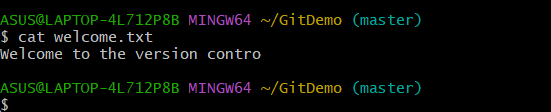
3. To create a file **“welcome.txt”** and add content to the file, execute the command



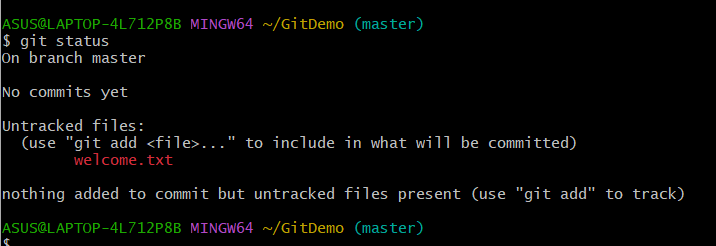
4. To verify if the file “welcome.txt” is created, execute



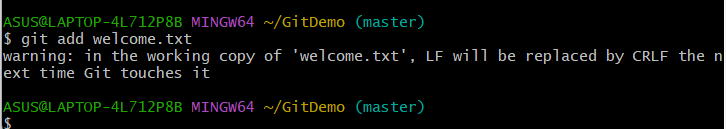
5. To verify the content, execute the command



6. Check the status by executing

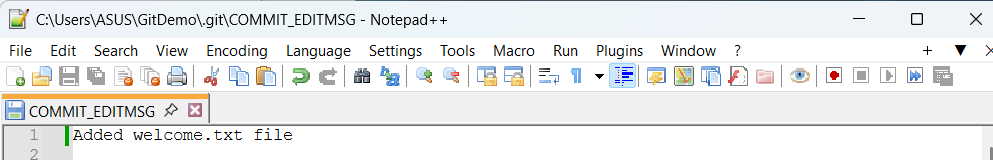


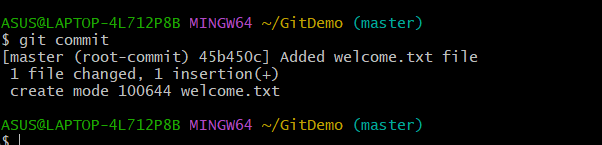
7. To make the file to be tracked by Git repository, execute the command



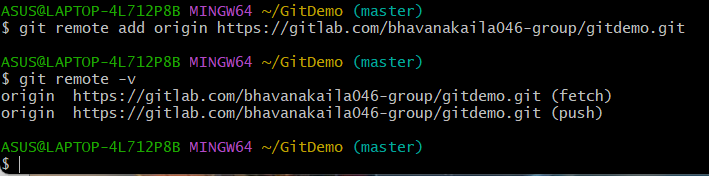
8. To add multi line comments, we are opening default editor to comment. Execute the command

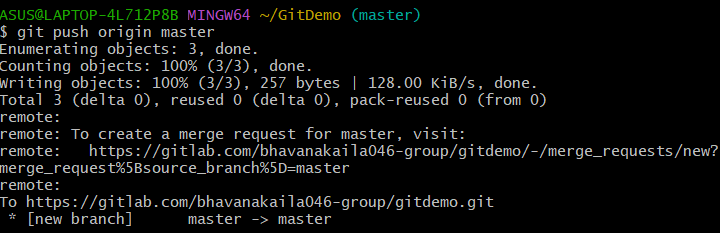
9. To check if local and “Working Directory” git repository are same, execute git status



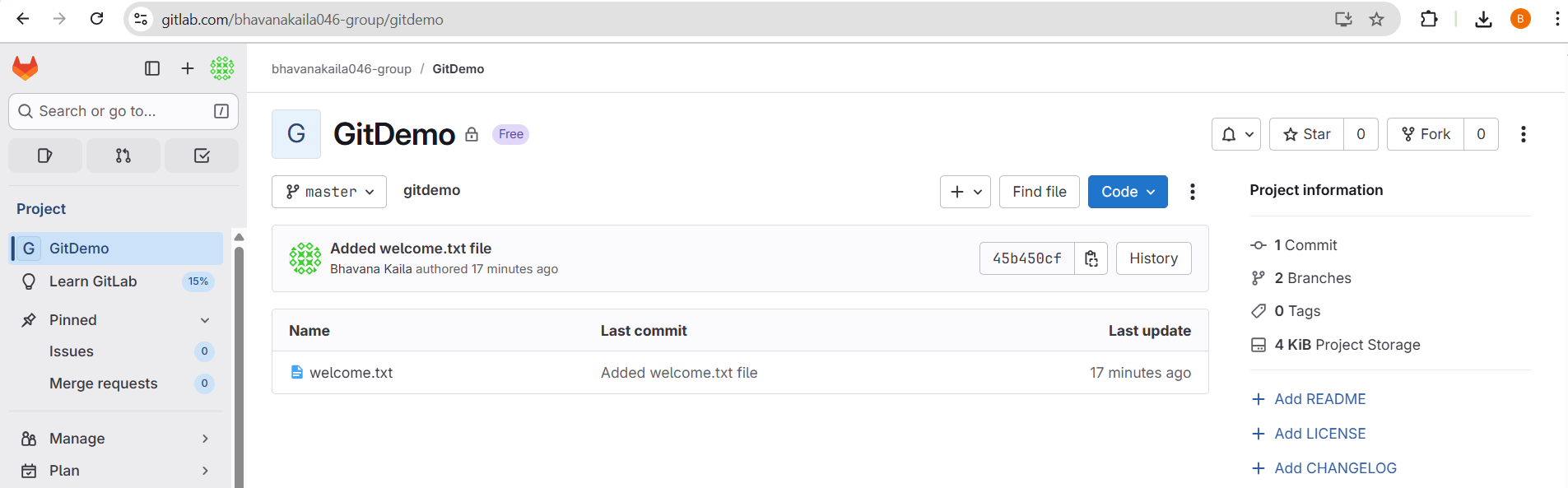


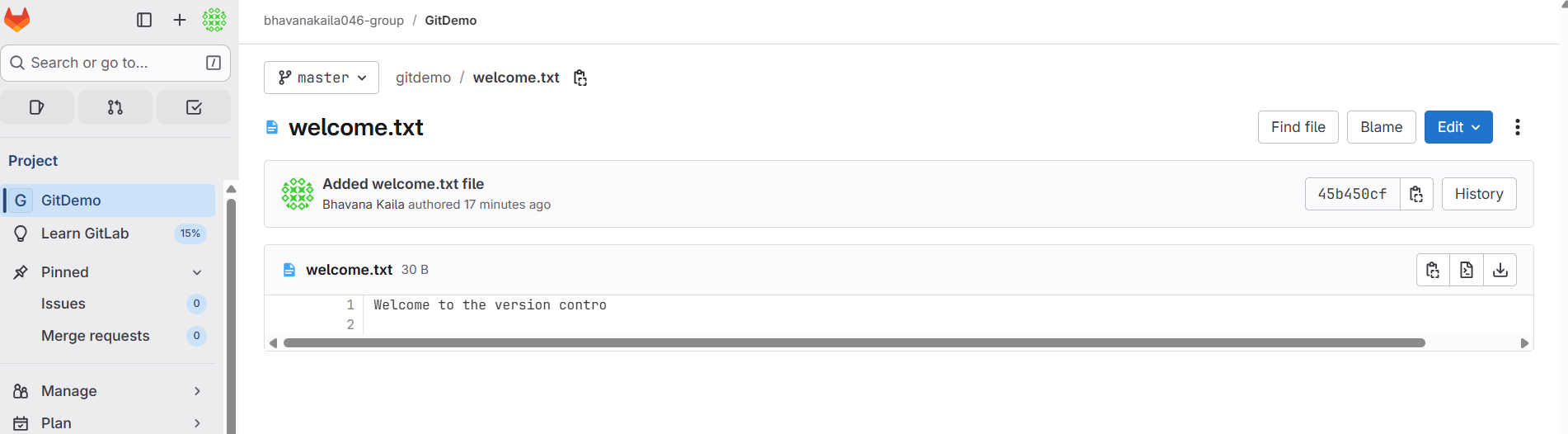
10. To push the local to remote repository, execute





welcome.txt will be added to the remote repository



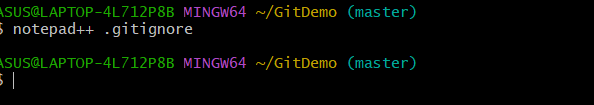


**2.Git-HOL:**

1. Navigate to GitDemo repository and Create a .log file and log directory

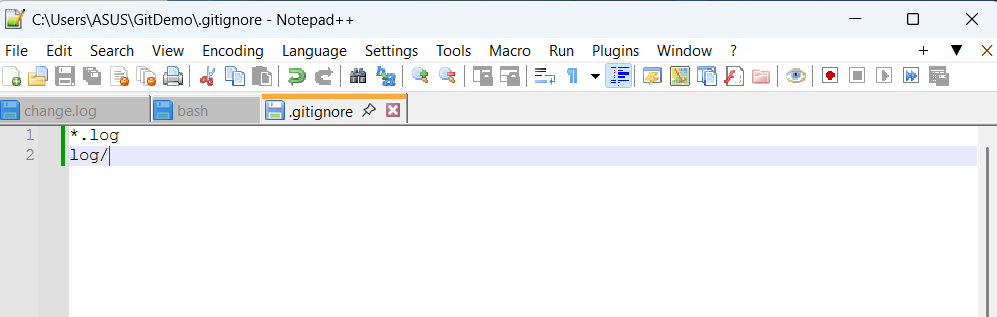


2. Create or update the .gitignore file

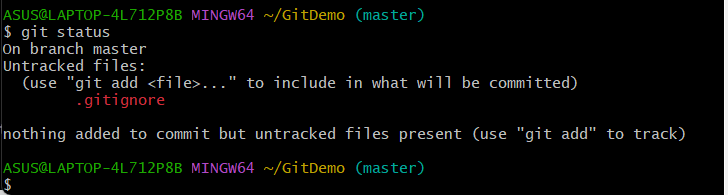


Ignore all files with .log extension.

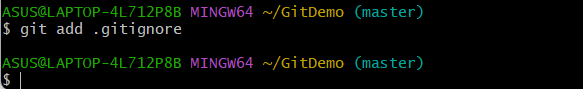
Ignore the entire log folder and its contents.



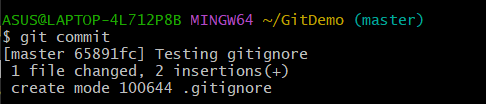
3. Verify the status using Git

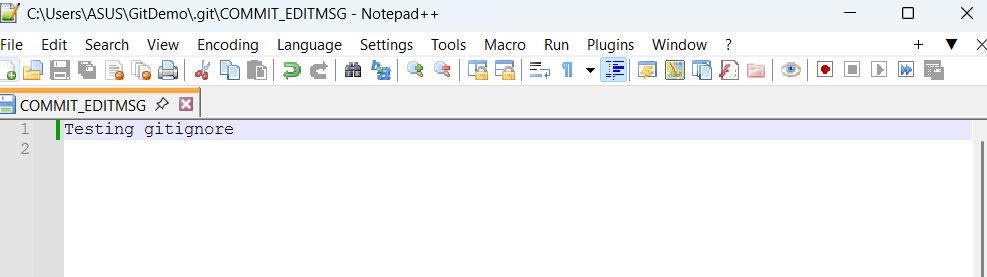


4. To make the file to be tracked by Git repository, execute the command

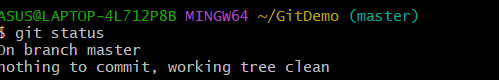


5. Commit the updated .gitignore

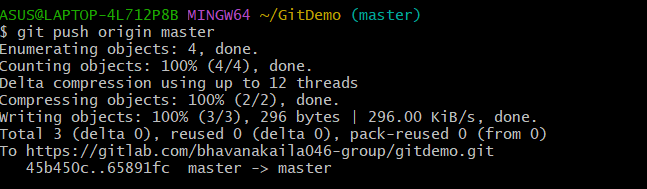




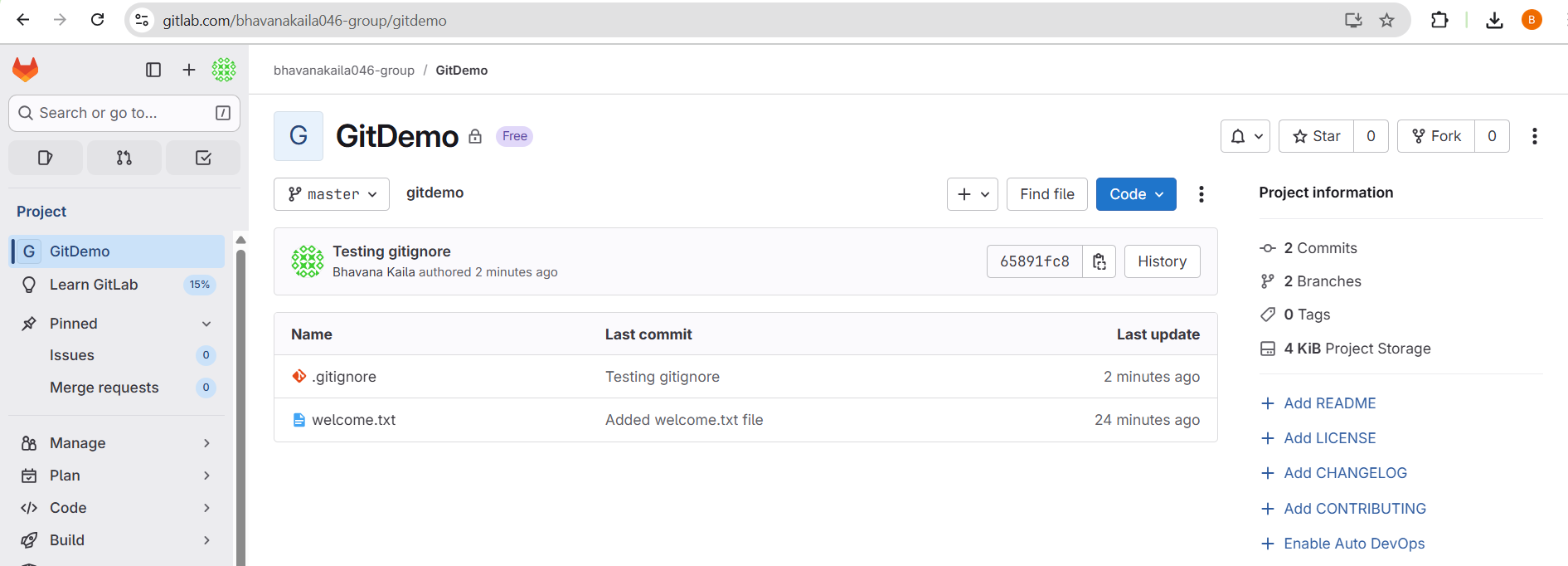
6. Execute git status

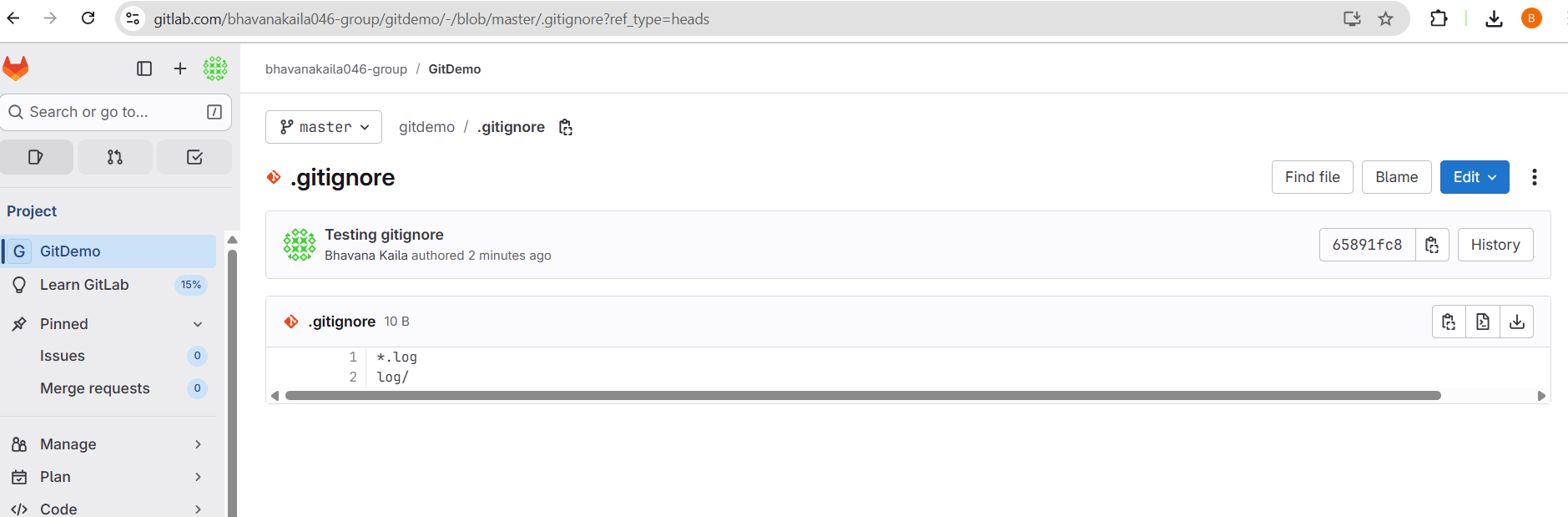


7. Push the commit to remote repository



.gitignore file is added to the remote repository and the files mentioned in .gitignore are ignored and not added.

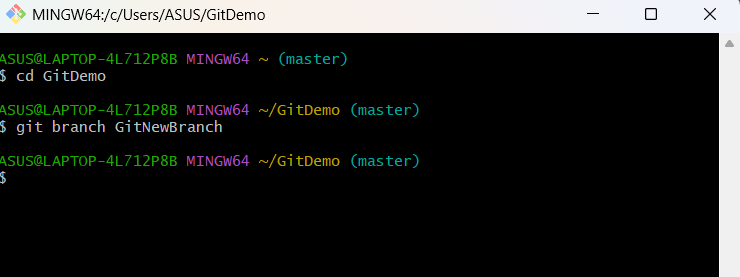




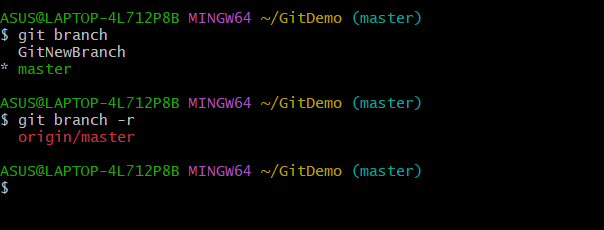
**3.Git-HOL:**

**Branching:**

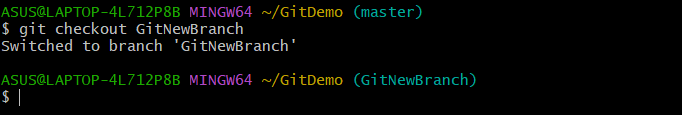
1. Navigate to GitDemo repository and Create a new branch named GitNewBranch



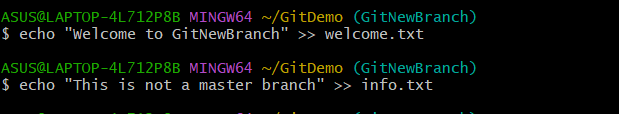
2. List all local and remote branches



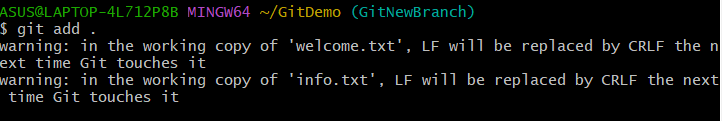
3. Switch to the newly created branch

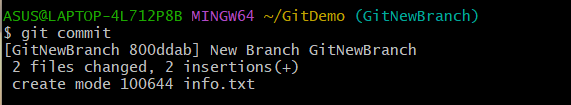


4. Add some files to the branch

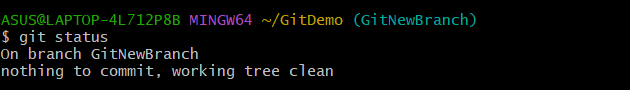


5. Commit the changes to the branch



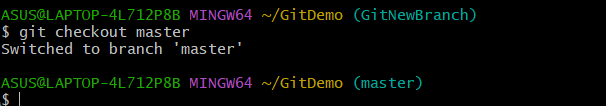


6. Check status

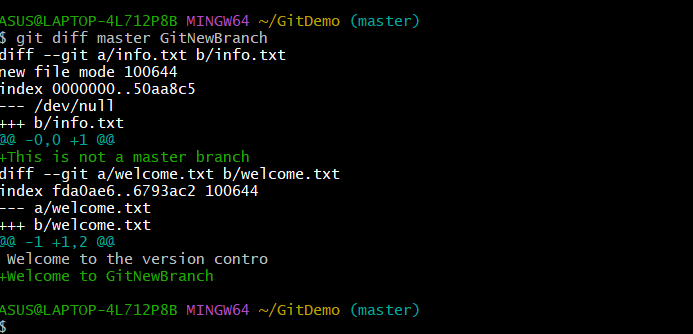


**Merging:**

1. Switch back to master

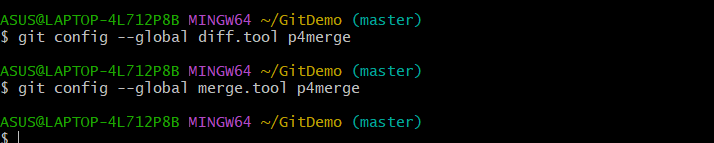


2. List out all the command-line differences between master and GitNewBranch

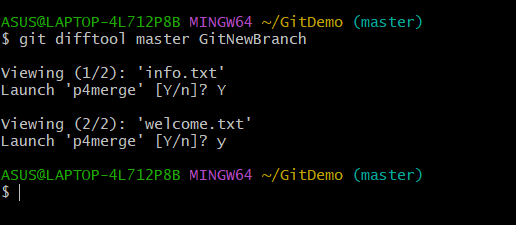


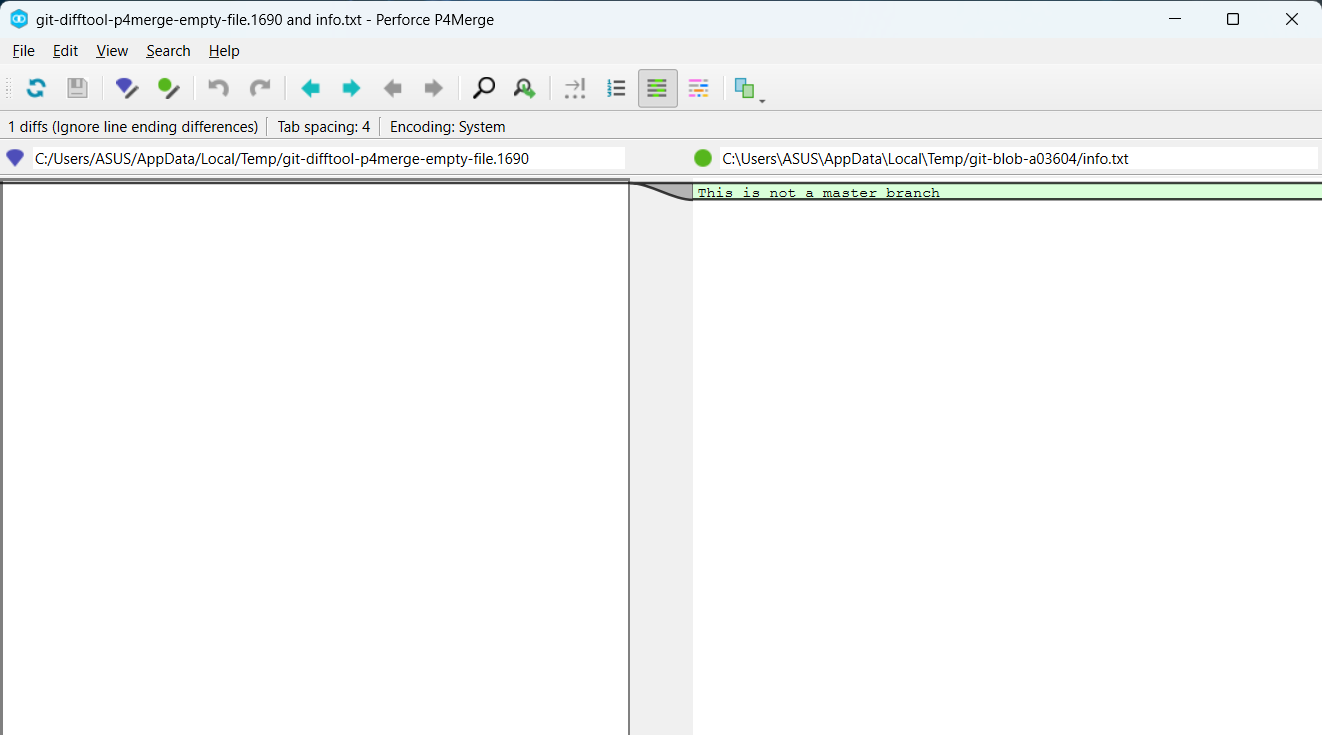
3. View visual differences using P4Merge

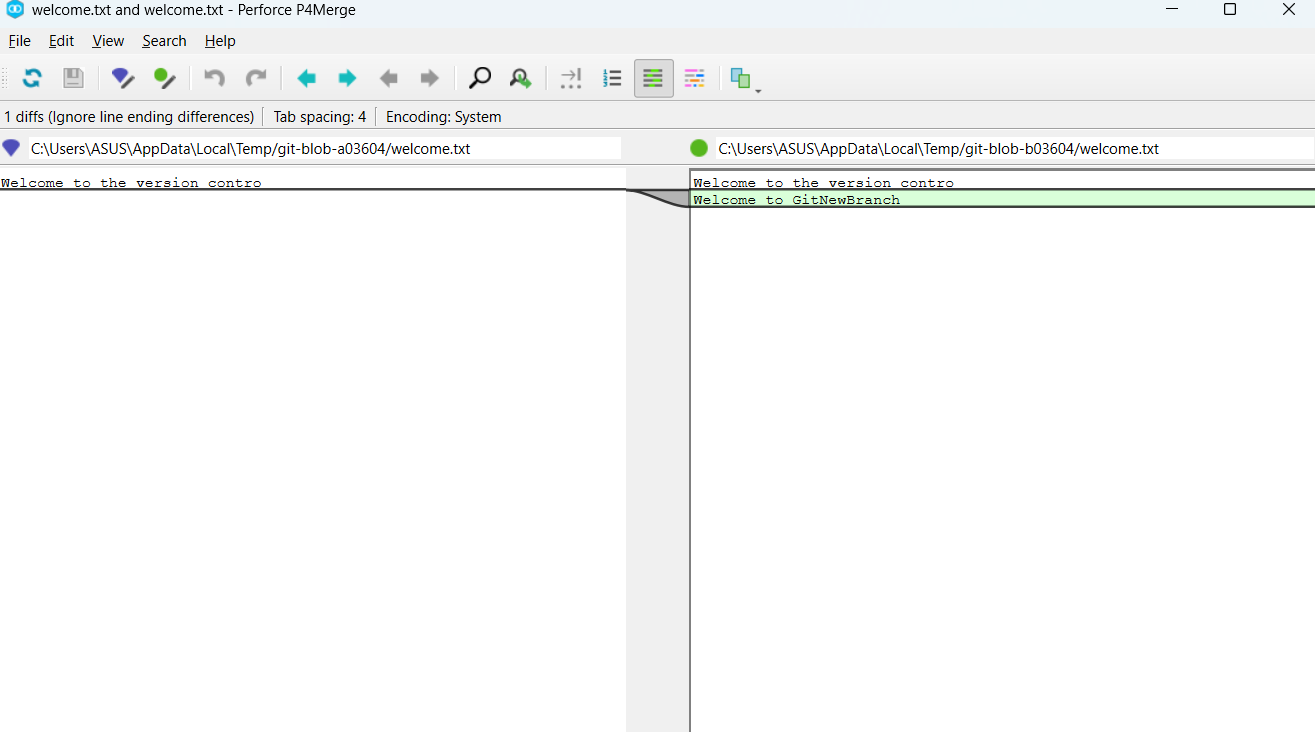
If P4Merge is not set, set it using:

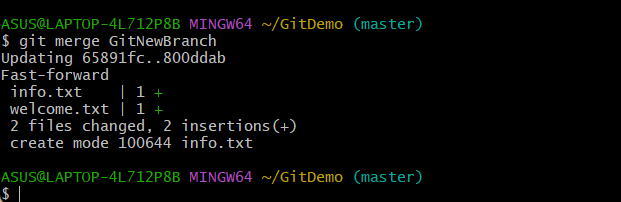


If P4Merge is installed and configured as the diff tool:

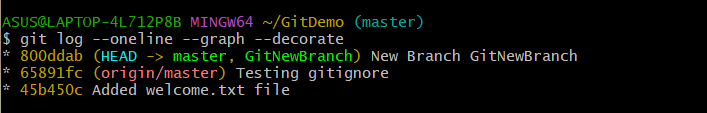




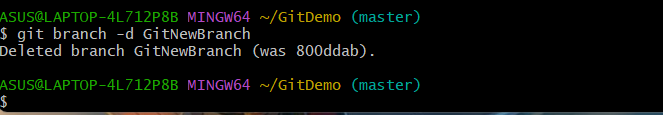
4. Merge GitNewBranch into master

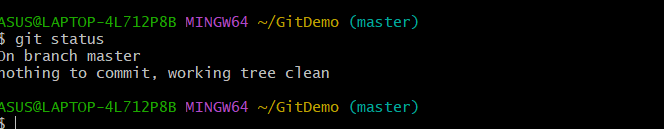


5. View Git log after merge



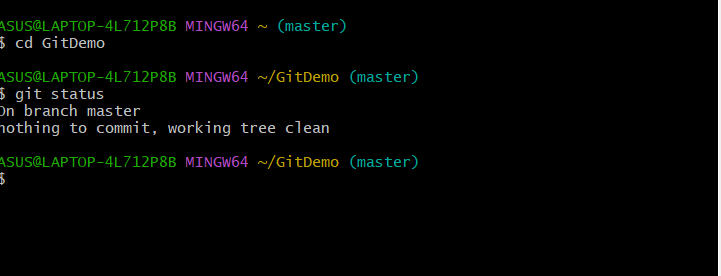
6. Delete the branch after merging and check status



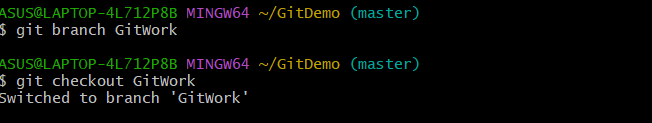


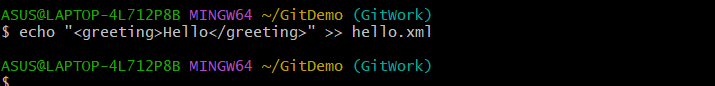
**4.Git-HOL:**

1.Verify if master is in clean state

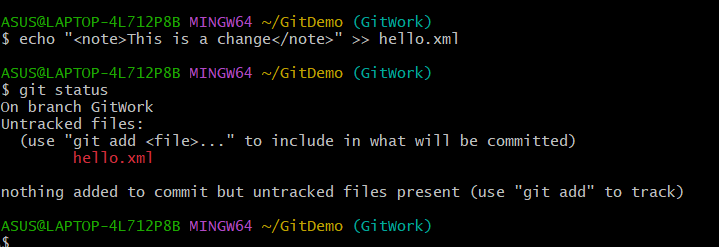


2. Create a branch GitWork and add a file hello.xml

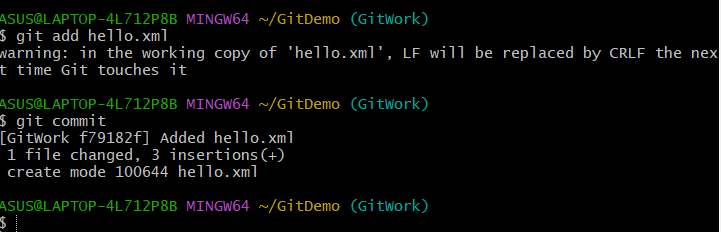


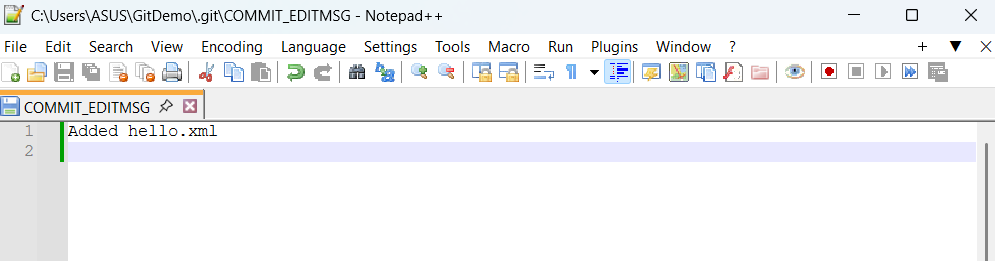


3. Update content of hello.xml and observe the status



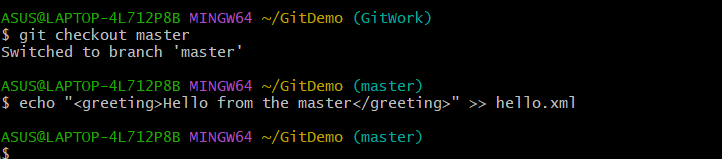
4. Commit the changes to reflect in the branch



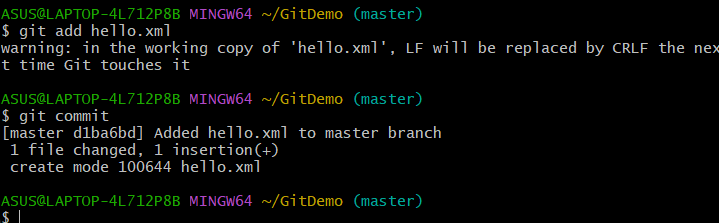


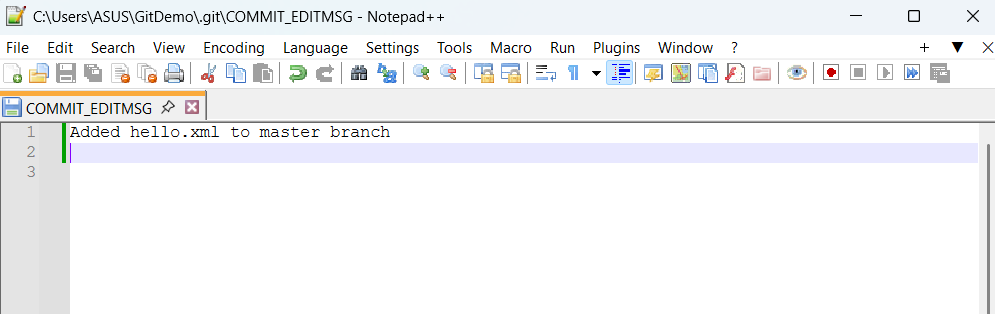
5. Switch to master

6. Add a file hello.xml in master with different content

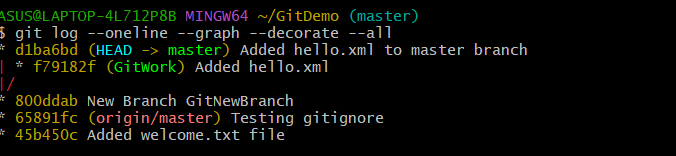


7. Commit the changes to master

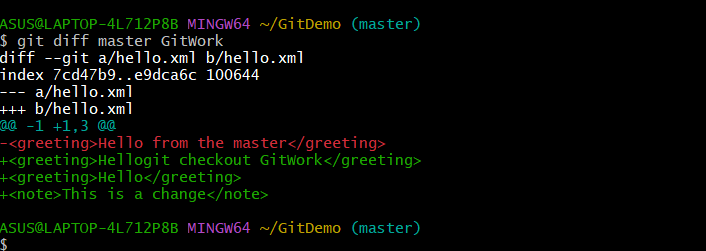




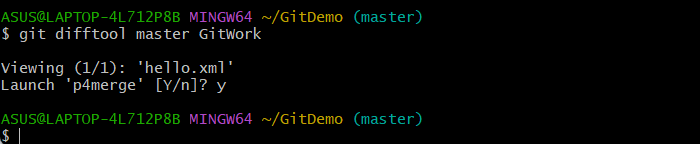
8. Observe the log

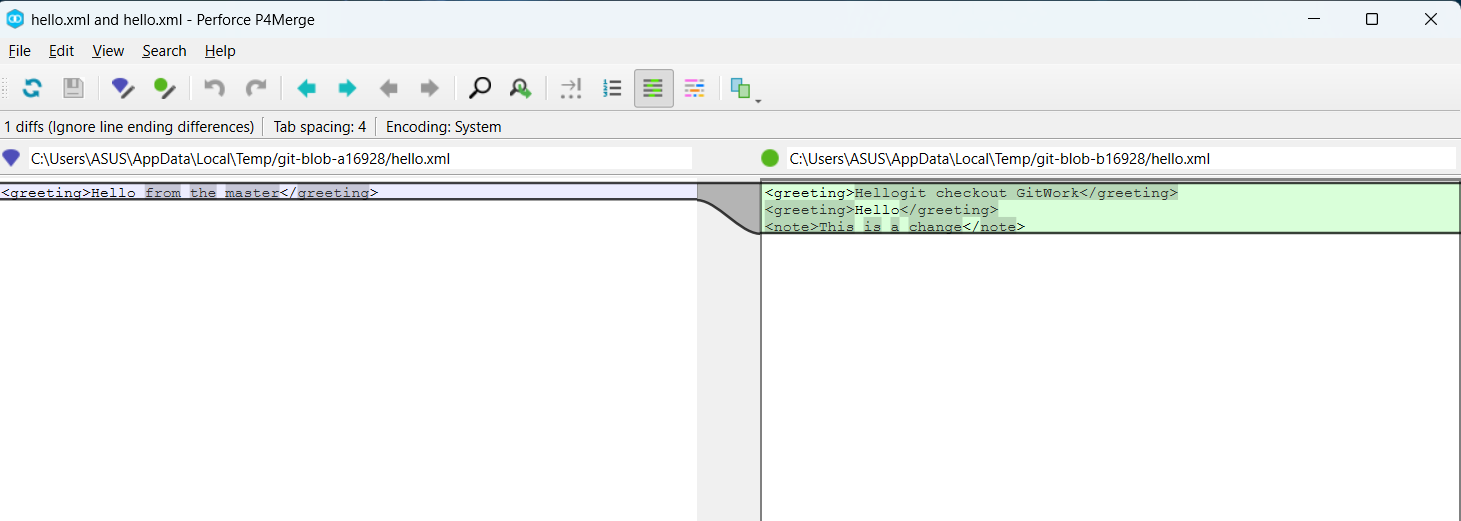


9. Check the differences with Git diff

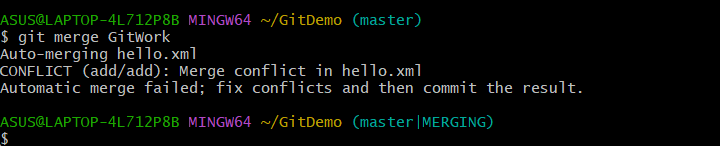


10. Use P4Merge tool to view differences

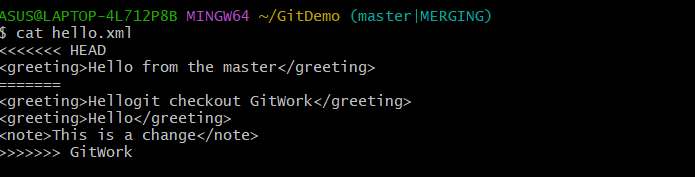




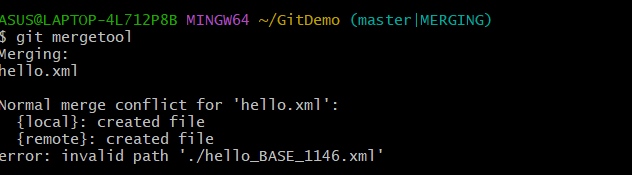
11. Merge the branch GitWork into master

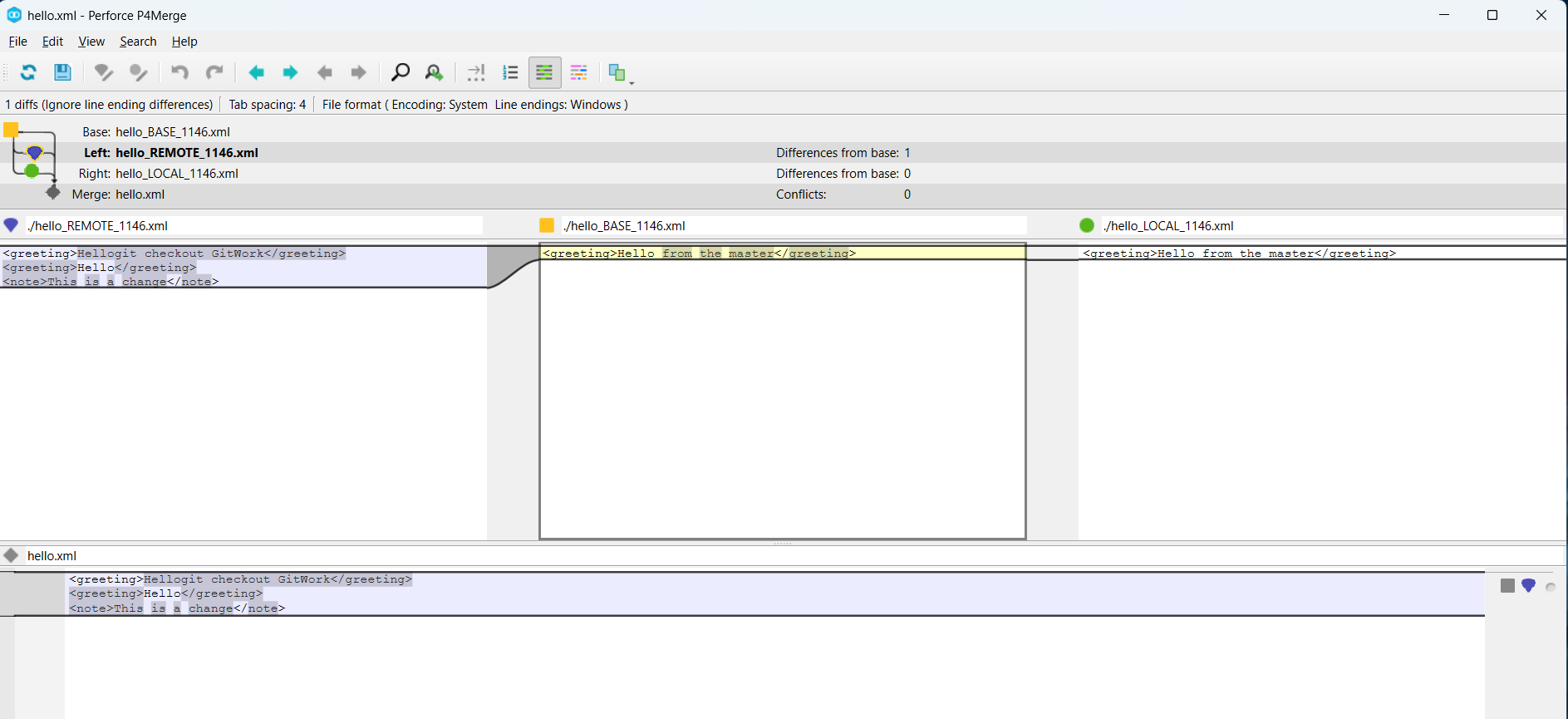


12. Observe the Git conflict markup

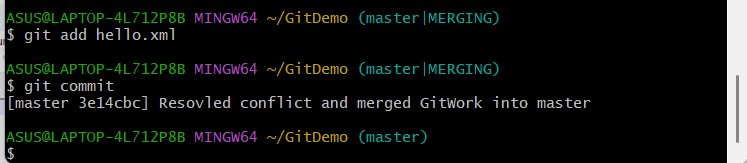


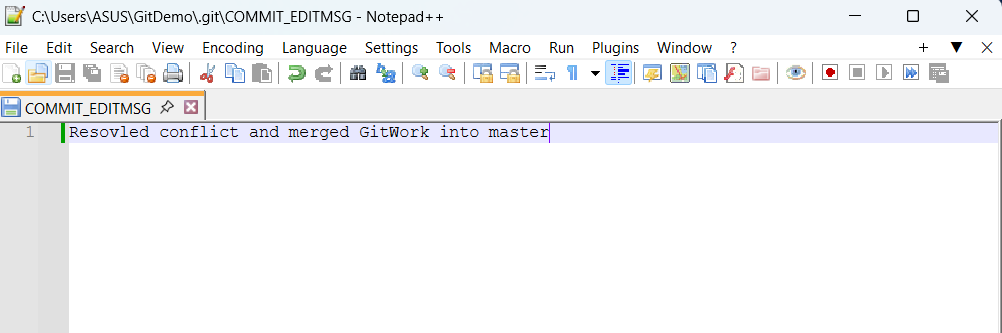
13. Use 3-way merge tool to resolve conflict



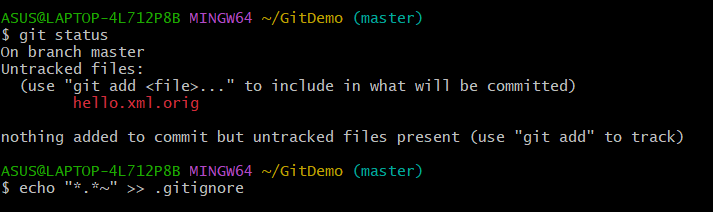


14. Commit the changes after resolving conflict

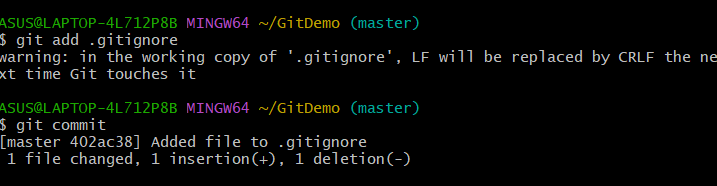


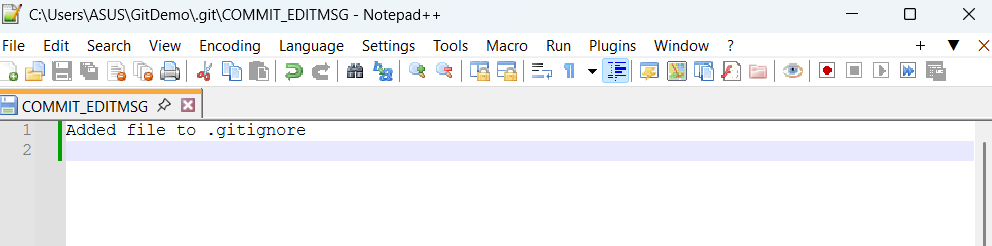


15. Observe Git status and add backup files to .gitignore

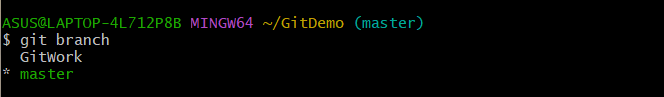


16. Commit the .gitignore changes

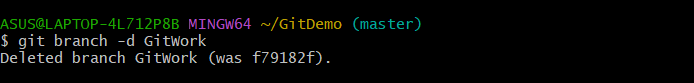




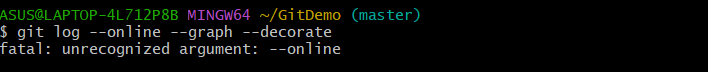
17. List all available branches



18. Delete the merged branch

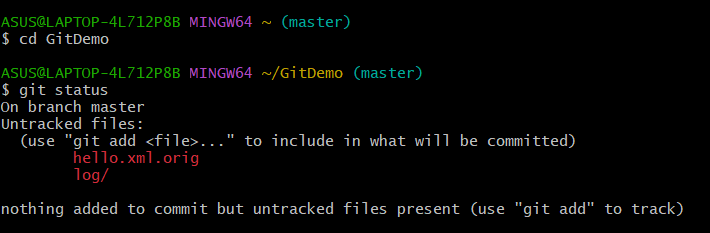


19. Final log to visualize history

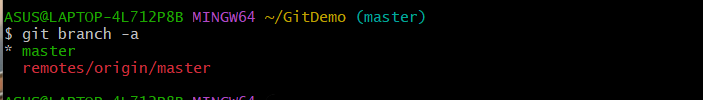


**5.Git-HOL:**

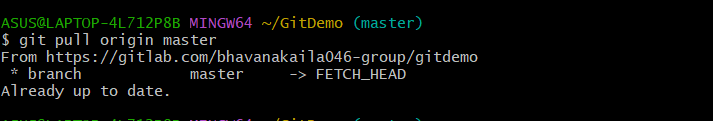
1. Verify if master is in clean state.



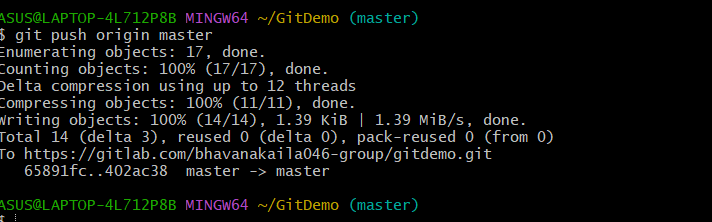
2. List out all the available branches.



3. Pull the remote git repository to the master



4. Push the changes, which are pending to the remote repository.



5. Observe if the changes are reflected in the remote repository.

