Exercise 1: Setting up GIT

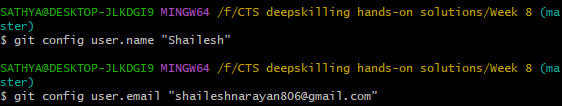
1. Create account in gitlab, then open the git bash and type in the command “git –version”



This implies that git is successfully installed.

2. Now I should configure it with my gitlab account, Since I already have github linked and I don’t want to override that, I will configure git to use gitlab credentials only for the “week 8” directory(repository)

3. After setting up SSH keys and initializing git on the week 8 folder, we add the config commands



4. After installing notepad++, set the environment variable for notepad++ then, we run the following commands to set up notepad++ as core editor for git



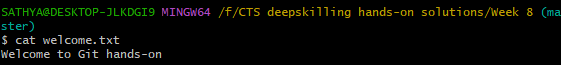
5. Check the setup using

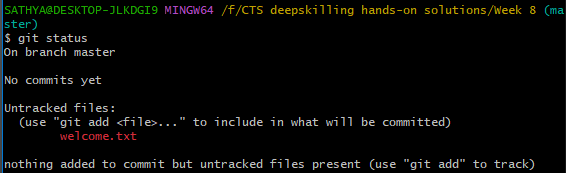


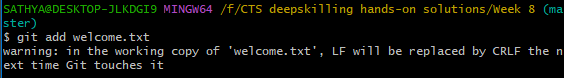
6. Now we create a new file Welcome.txt in the week 8 folder



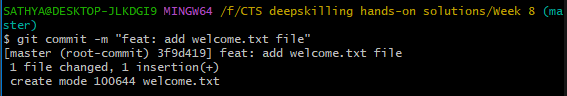
7. Verify the created file using “cat” command



8. Check the status of the file  


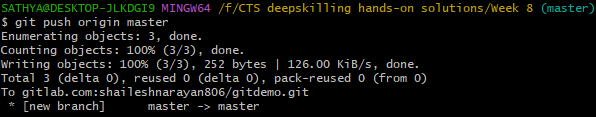
9. Use the “git add” command to stage the file  


10. Commit the staged file using “git commit” command

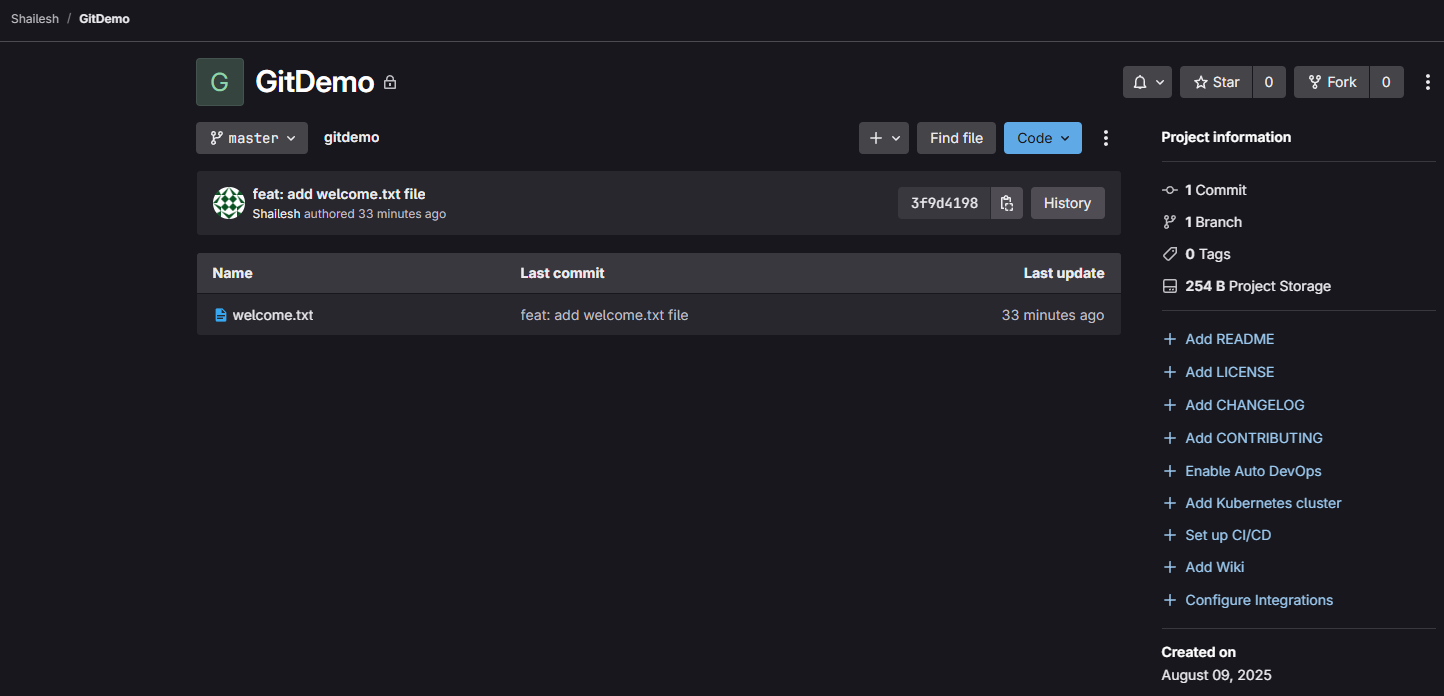


11. Now connect the remote repository on gitlab and push the changes





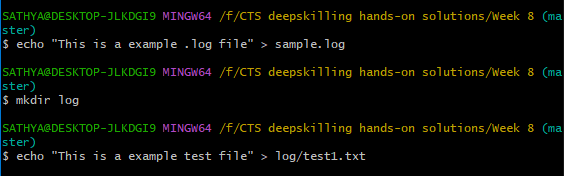
**OUTPUT(gitlab web interface)**



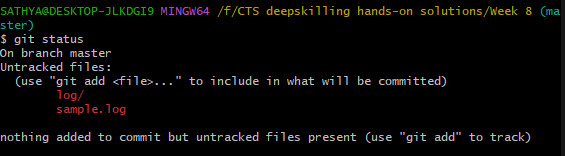
Exercise 2: Creating .gitignore file to keep .log files ignored

1. We open the git bash in the “Week 8” (folder where we previously initialized git) folder.

2. Create dummy log file called “sample.log” and a folder named “log” inside which we create a dummy test file.

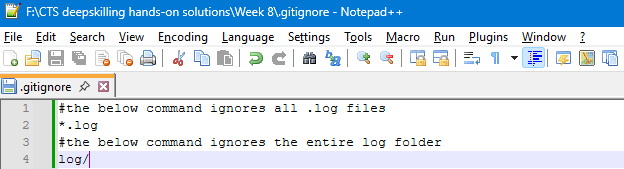


3. Now run the command git status to check current status of the files

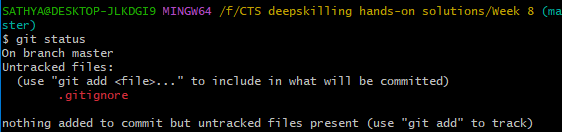


as we can see it is showing the log files and log folder as untracked, but we want those files to be ignored.

4. We now create “.gitignore” file using notepad++ , to ignore the .log files and log folder

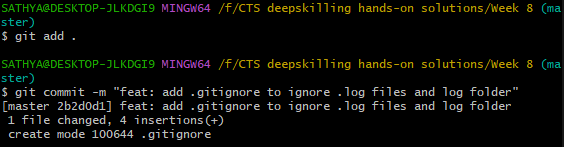


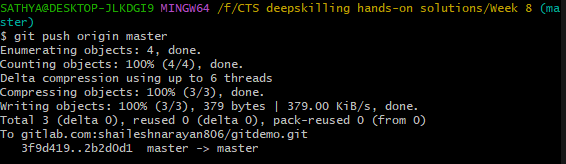
save this file and run the git status again to make sure whether the .log files and log folder is ignored.



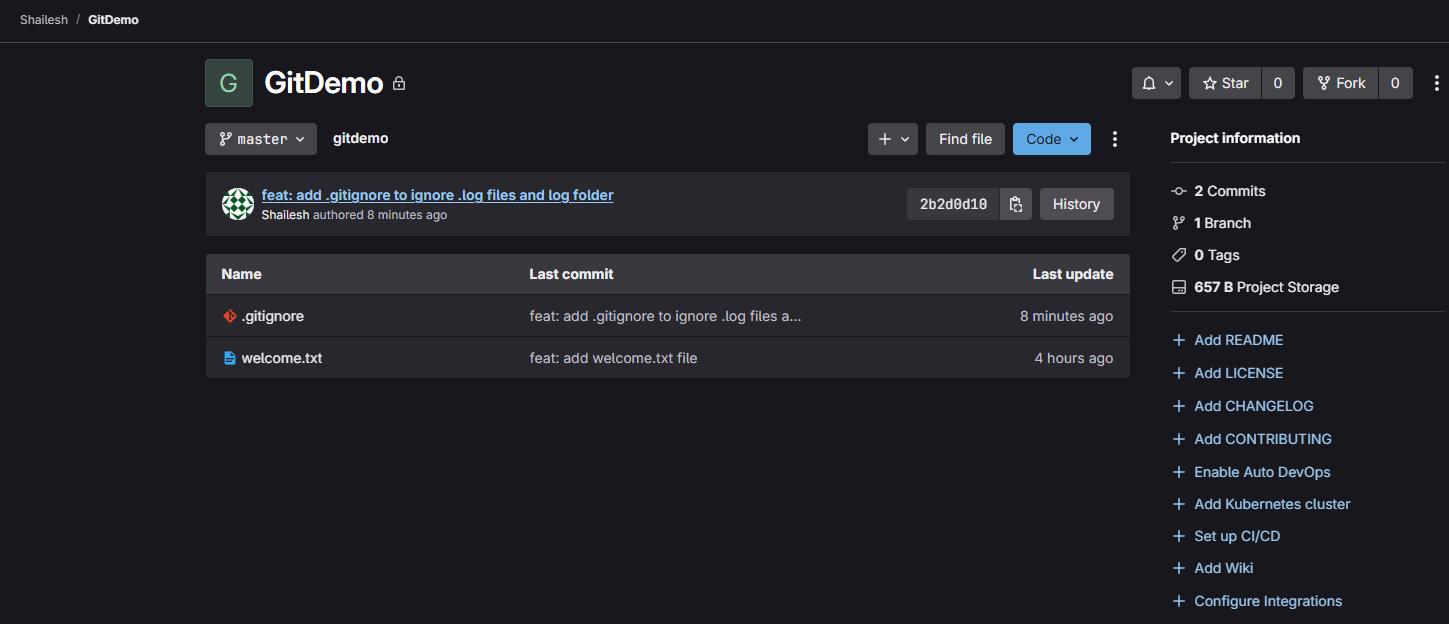
5. As we can see the .log files and log folder is not shown here, meaning the “.gitignore” files is working as expected

6. Now we will just add and commit these changes





**OUTPUT(gitlab web interface)**

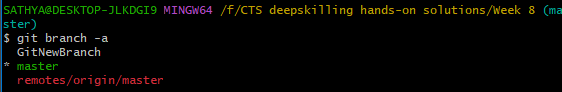


Exercise 3: Setting up git with p4 merge tool

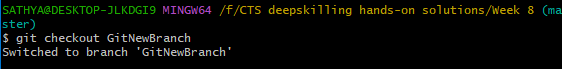
1. We create a new branch by the following command



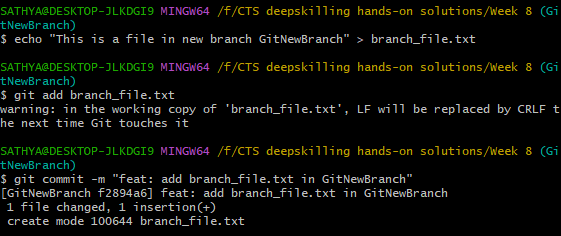
2. Check the branch that we are currently in



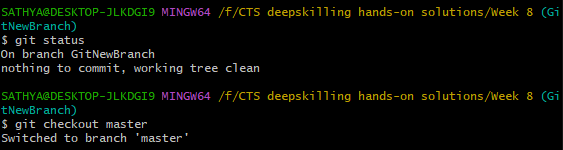
3. Switch to the newly created branch

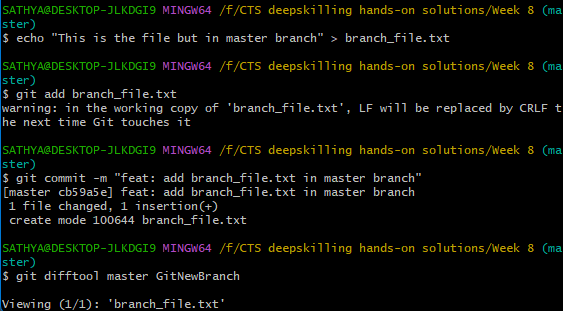


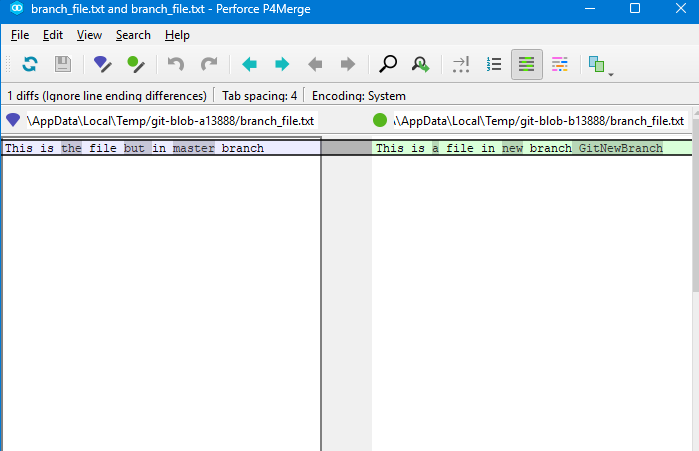
4. Now create a new dummy file in this branch



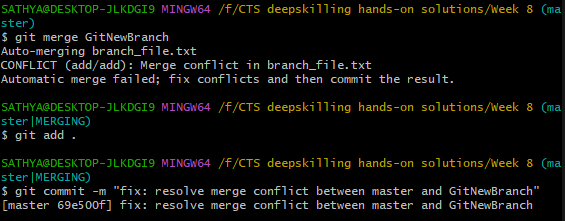
5. Now we will switch back to master branch and create the same file in master branch so that we can see the difference in “p4merge tool”



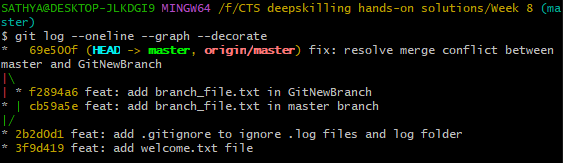




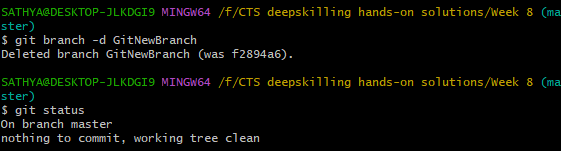
6. Now if we try and merge it will raise a conflict and we will have to resolve it using notepad++ just like the below screenshots



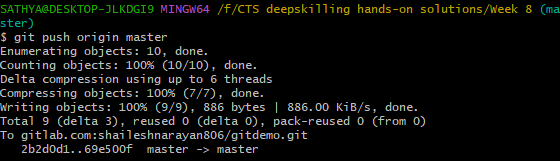
Using the command “git log --oneline --graph –decorate” we can visualize the git history



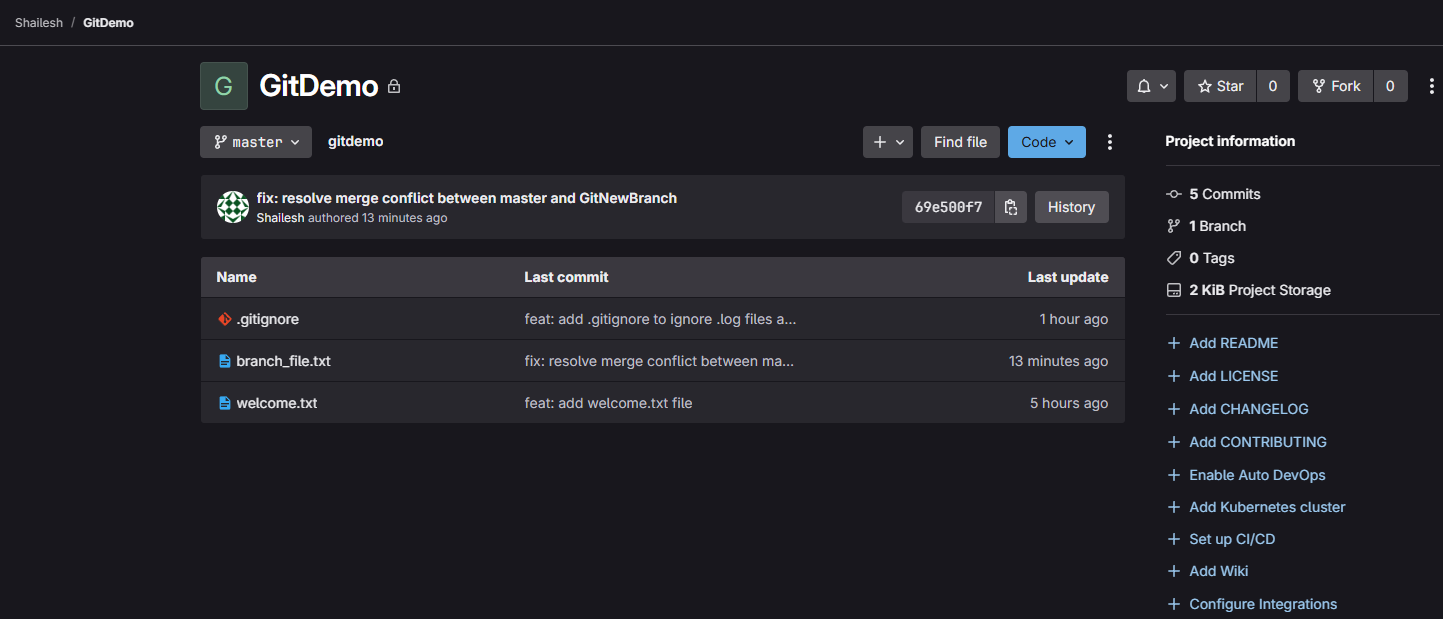
7. Now we will just delete the newly created branch and finish up this hands-on



8 . We finished with pushing all the changes to remote repo

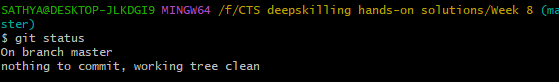


**OUTPUT(gitlab web interface)**

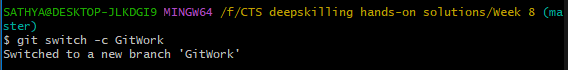
****

Exercise 4: Git merging and handling git conflicts

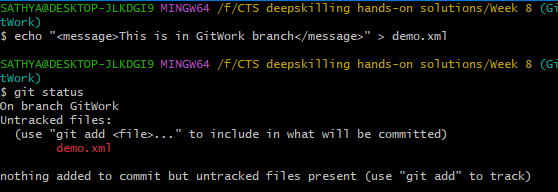
1. First check the master branch using “git status” command

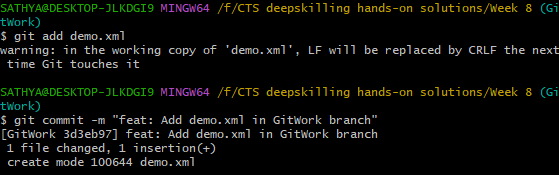


2. Create a new branch “GitWork” and switch to it

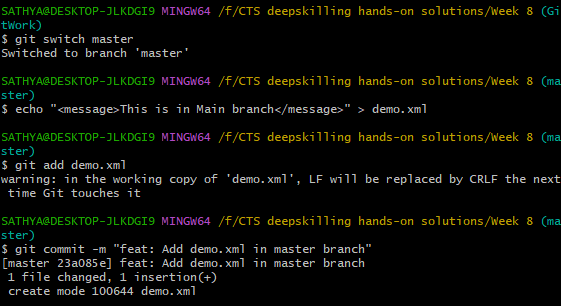


3. Create a dummy .xml file named “demo.xml”, check git status, add and commit it

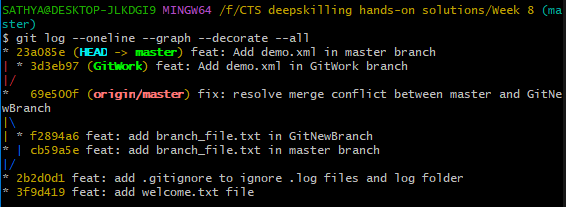




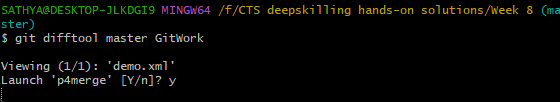
4. Now switch back to master branch and create a .xml file with same name but different content in it

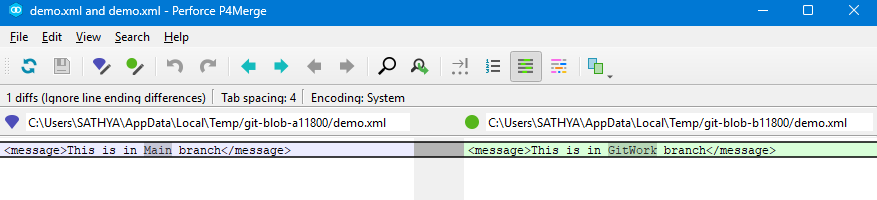


5. Now visualize the git history so far

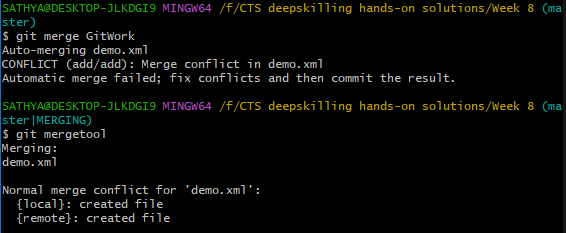


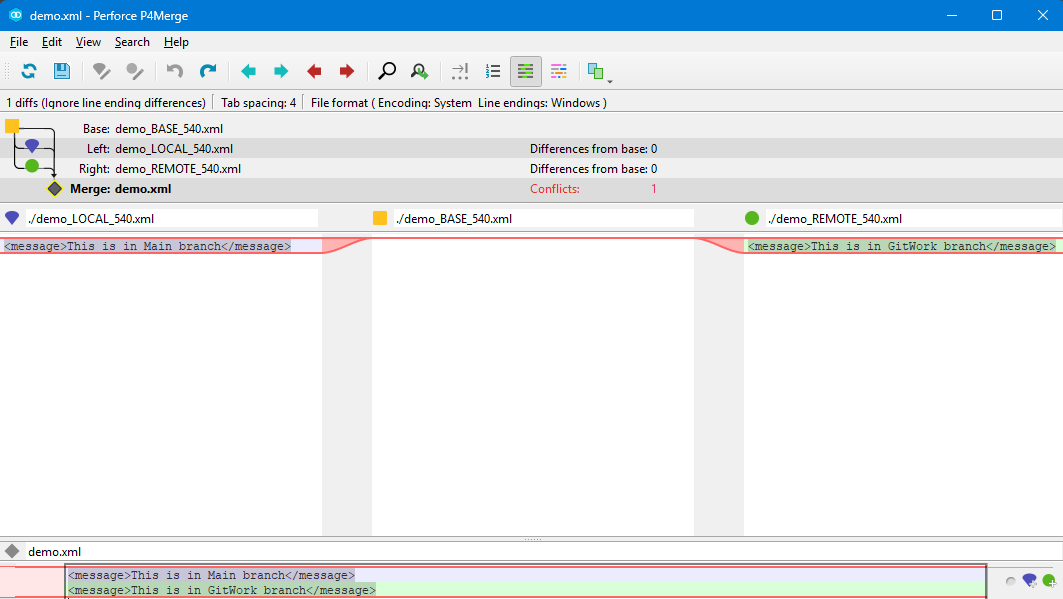
6. Using p4merge check the differences in the two files



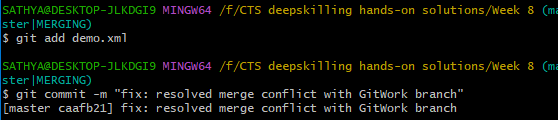


7. Now try and merge it, it will result in conflict and we will handle in using p4merge tool

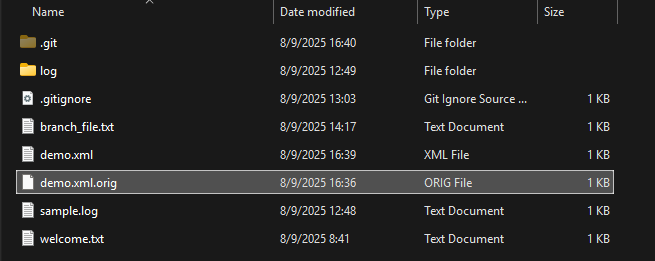


****

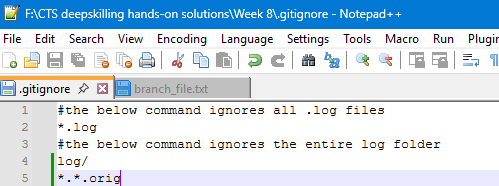
8. Now we make changes and manually fix the conflict by keeping the content we want



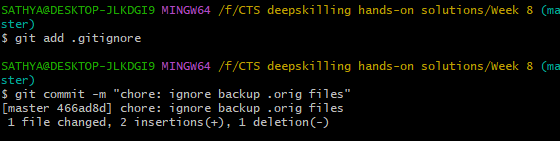
Now we have to make sure that the backup files are ignored..



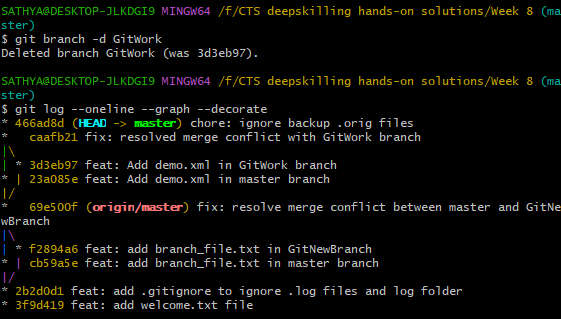
9. Make changes to the .gitignore using notepad++



10. Add and commit the .gitignore file

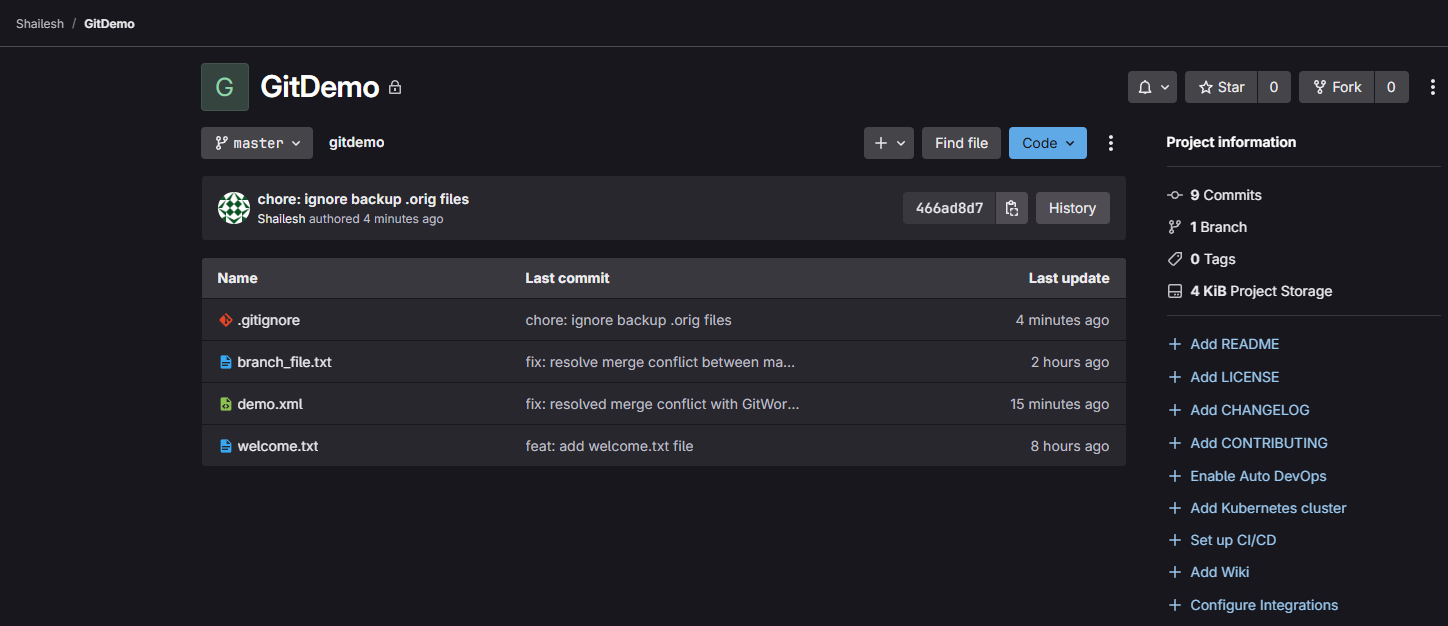


11. Now we delete the GitWork branch and view the log



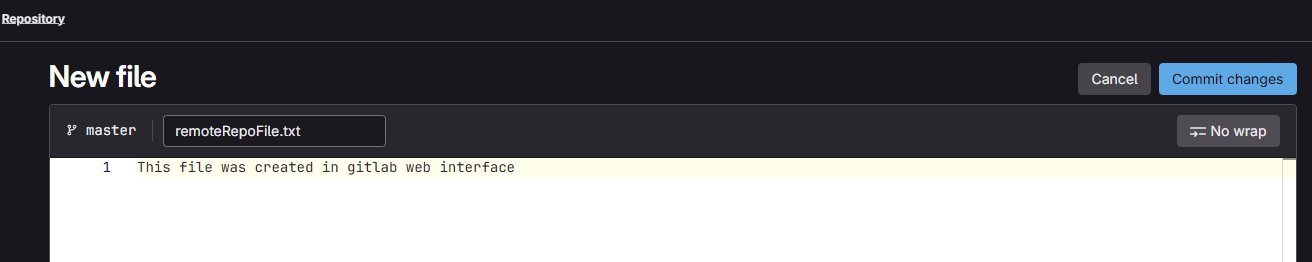
12. Finally we will push all these changes to remote repo

**OUTPUT(gitlab web interface)**

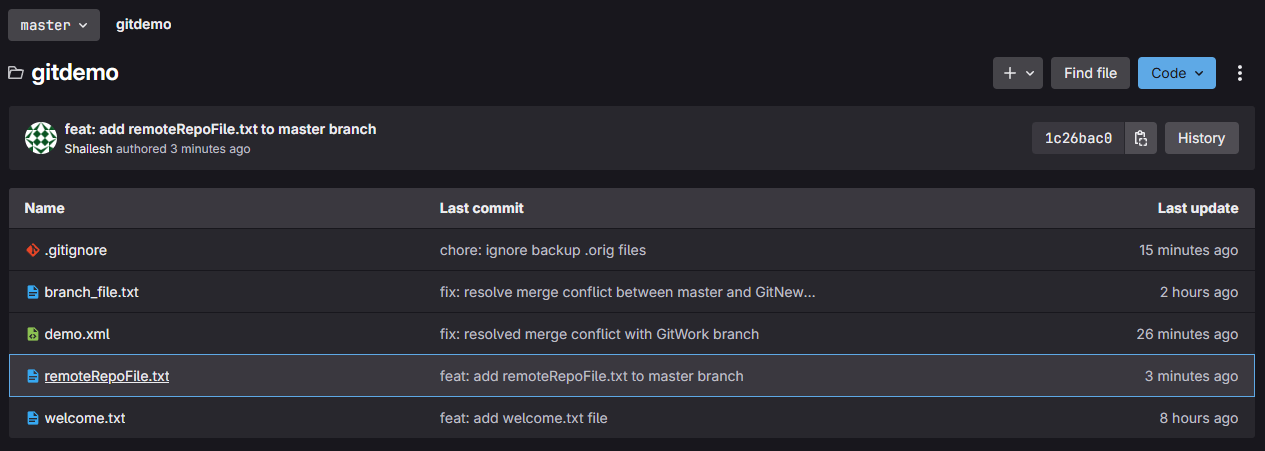


Exercise 5: Git push/pull requests from Local/Remote repositories

1. First we open gitlab, add a new file and commit to the master branch

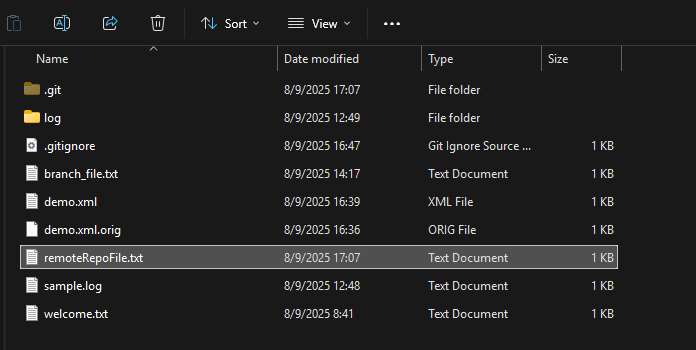


2. Now the new file is added in our remote repo



3. We will pull this changes from our local repo now



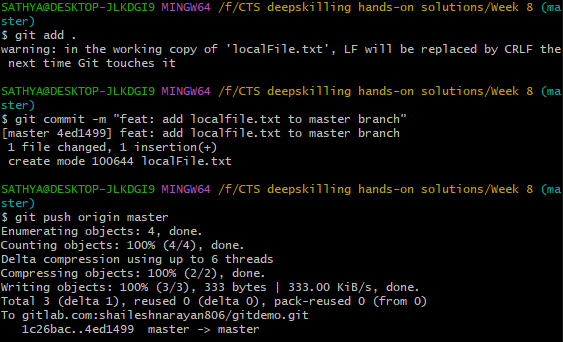


We can see that the remote file we created from gitlab web interface is fetched and added to the local repo

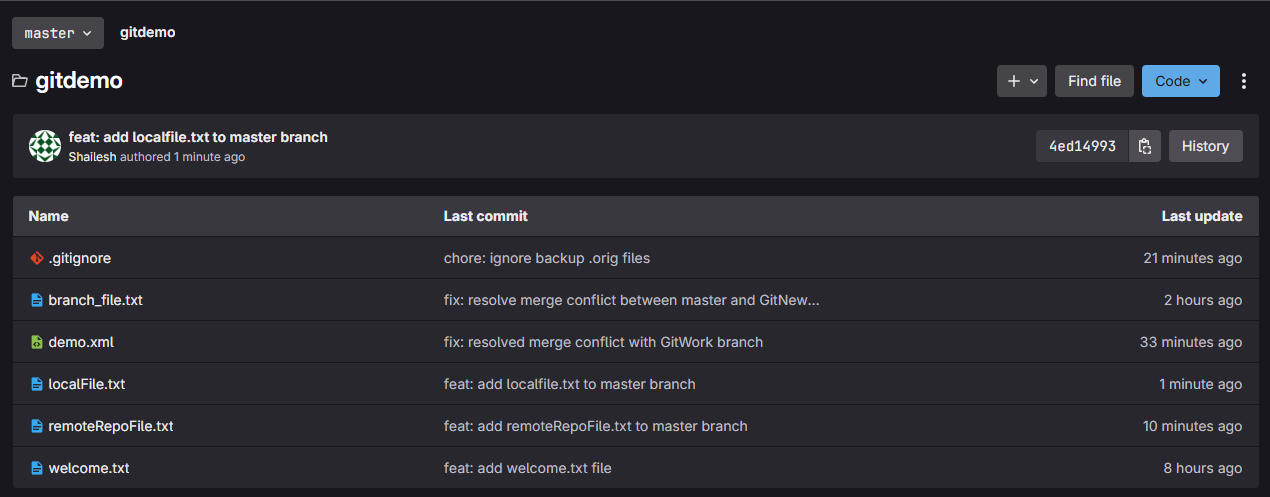
4. Now we will create some file in local repo and push the changes to remote repo and see how it is reflected in the web interface



5. Now add, commit and push the changes to the remote repo



**OUTPUT(gitlab web interface)**



We can see that the local file is now added to the remote repo.