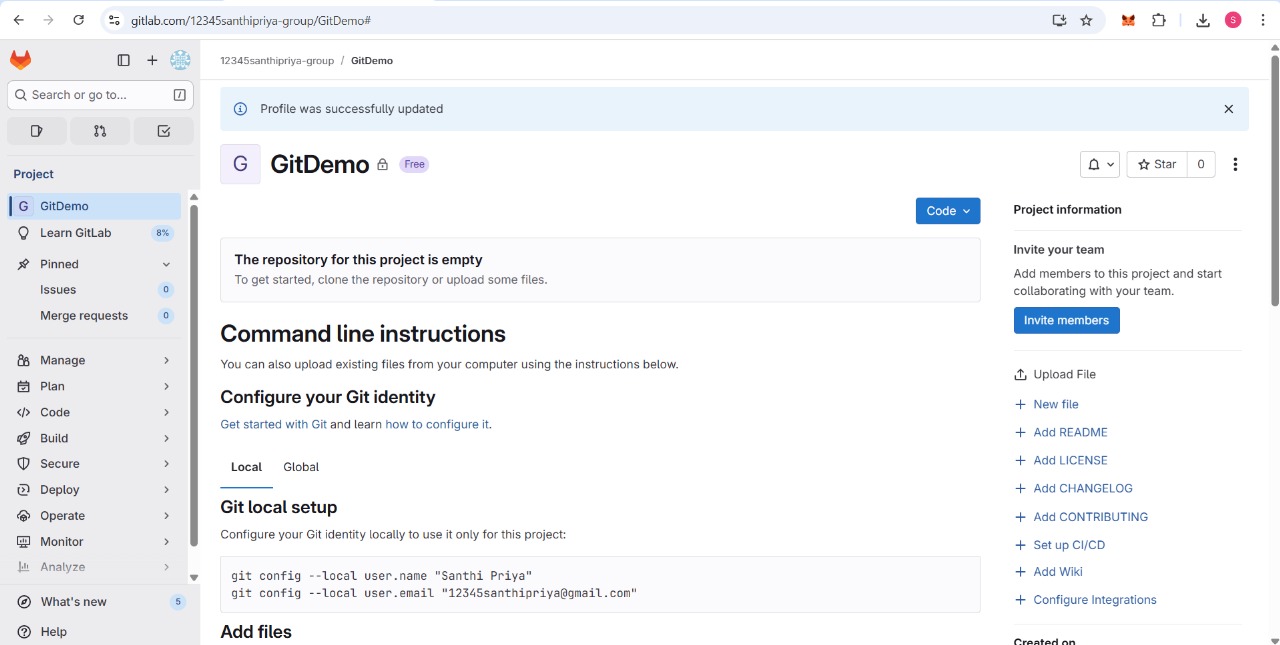
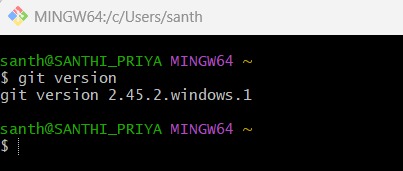
1. **GIT-HOL**

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

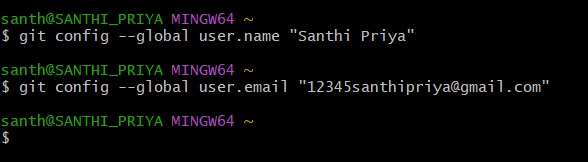
1. Signup with GitLab and register your credentials. Login to GitLab and create a “GitDemo” project.



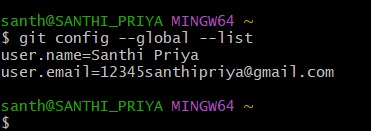
To check if Git client is installed properly: Open Git bash shell and execute



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

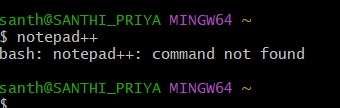


3. To check if the configuration is properly set, execute the following command



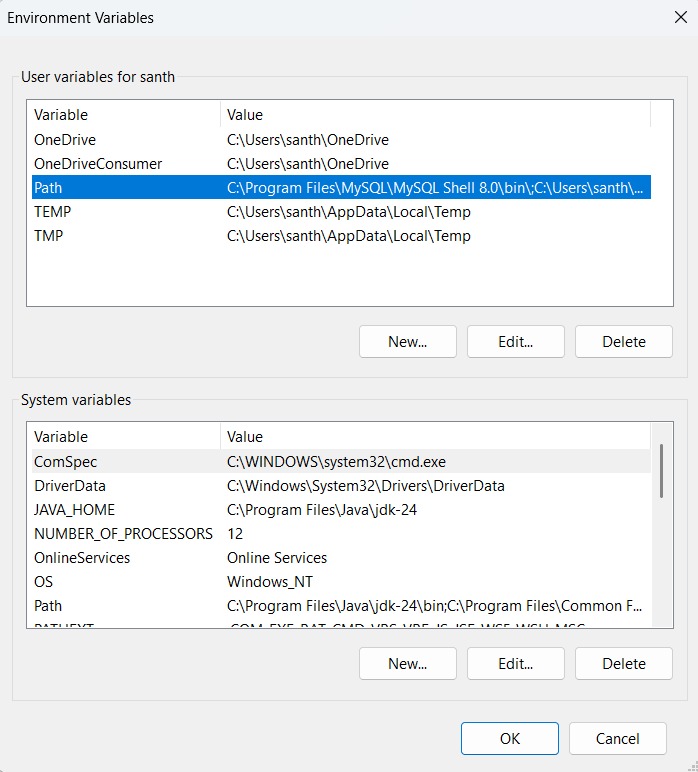
**Step 2: Integrate notepad++.exe to Git and make it a default editor**

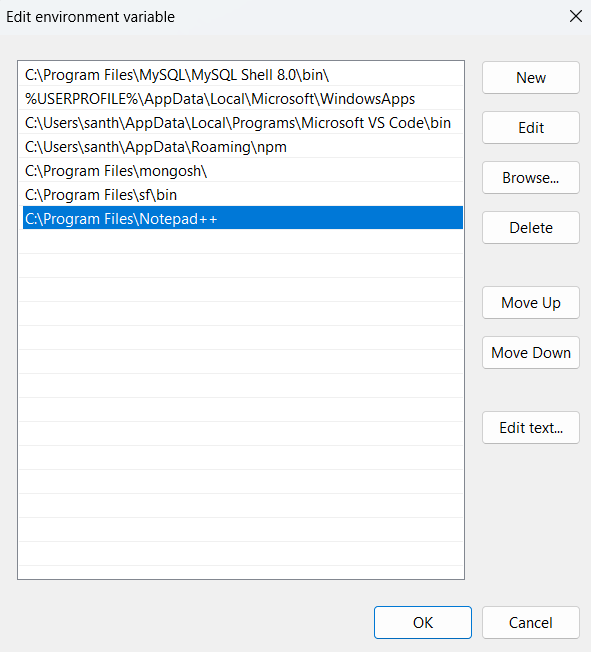
1.To check, if notepad++.exe execute from Git bash



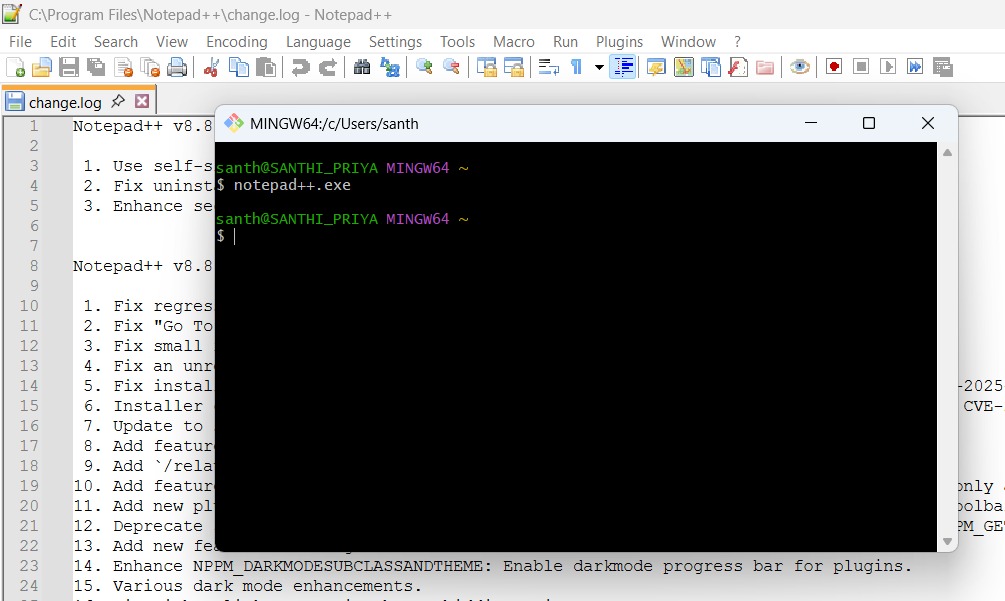
2. If Git bash could not able to recognize notepad++ command that implies notepad++.exe is note added to the environment path variable.

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”

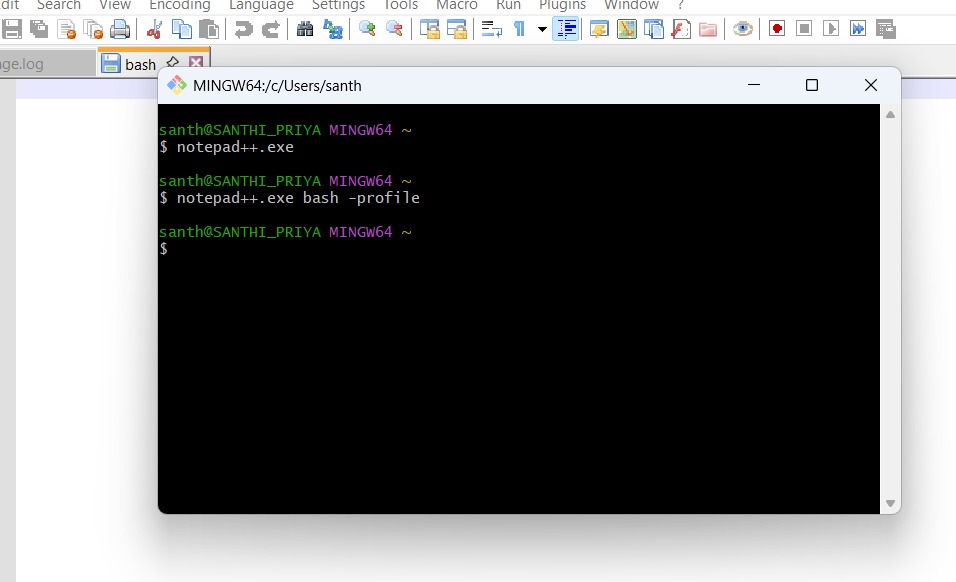




3. Exit Git bash shell, open bash shell and execute “notepad++.exe”, Now, notepad++ will open from Git bash shell



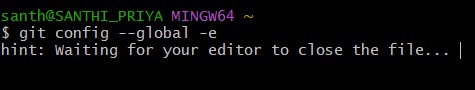
4. To create an alias command for notepad++.exe, execute “notepad++.exe bash -profile”, It will open notepad++ from bash shell, and create a user profile by adding the line in notepad++

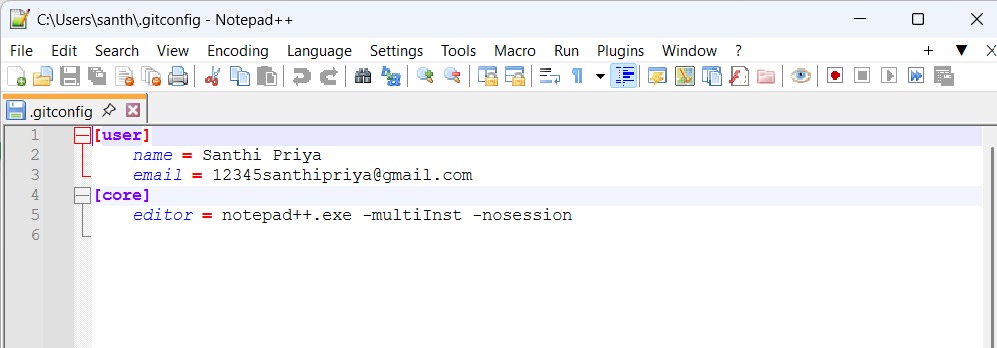


5. To configure the editor, execute the command



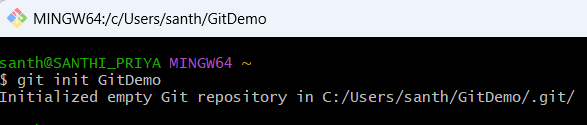
6. To verify if notepad++ is the default editor, execute the command





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

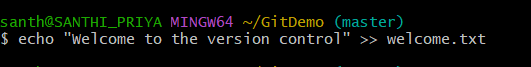
1.Open Git bash shell and create a new project “**GitDemo**” by executing the command



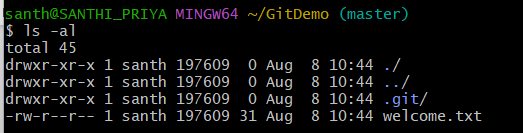
2. Git bash initializes the “**GitDemo**” repository. To verify, execute the command



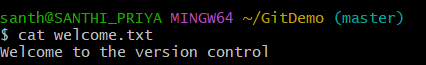
3. To create a file **“welcome.txt”** and add content to the file, execute the command



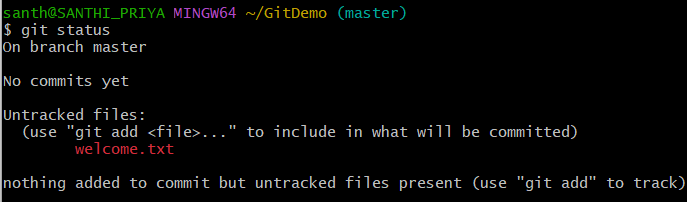
4. To verify if the file “welcome.txt” is created, execute



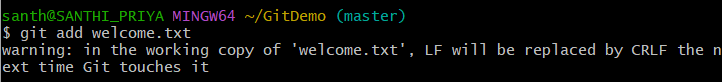
5. To verify the content, execute the command



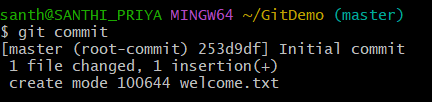
6. Check the status by executing



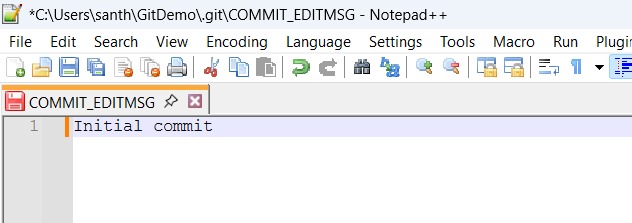
7. To make the file to be tracked by Git repository, execute the command



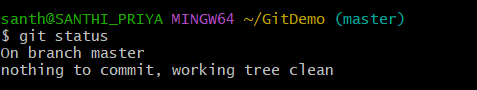
8. To add multi line comments, we are opening default editor to comment. Execute the command



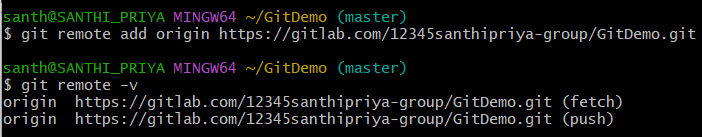
Notepad++ editor will open and to add multi-line comment with default editor

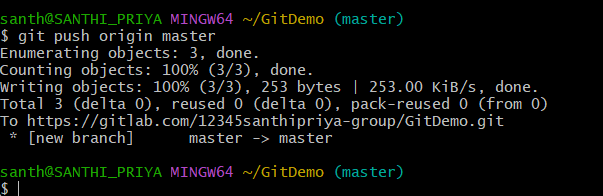


9. To check if local and “Working Directory” git repository are same, execute git status



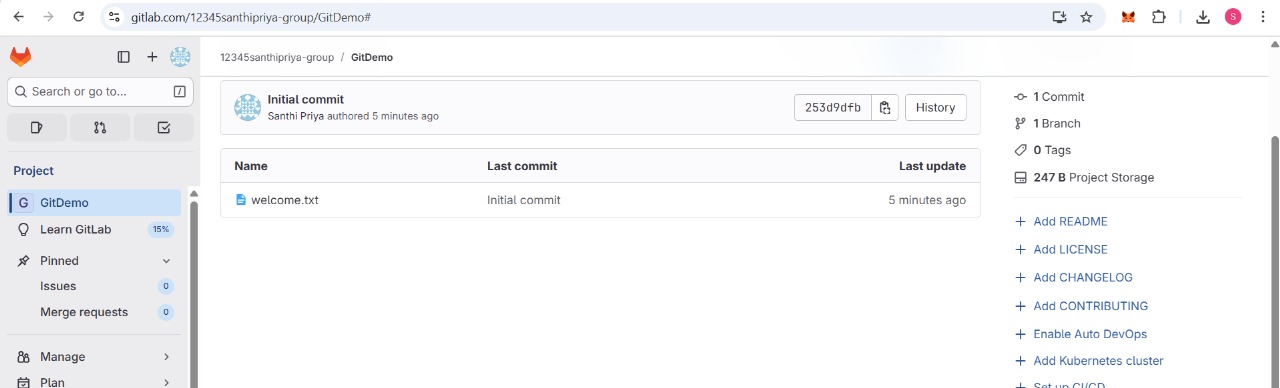
10. To push the local to remote repository, execute

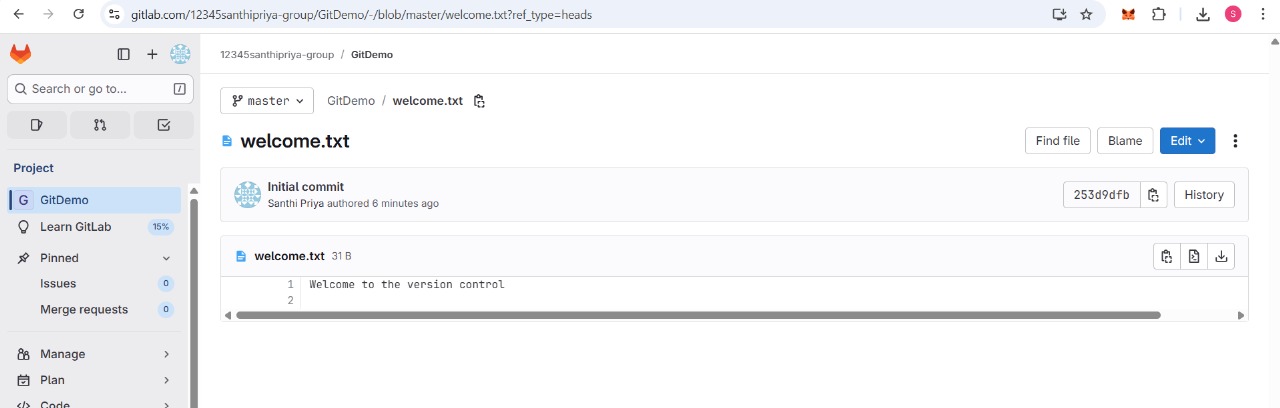




Authenticate the GitLab account in browser.

welcome.txt will be added to the remote repository



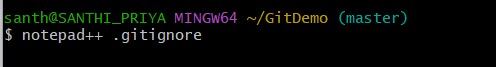


1. **GIT-HOL**

1. Navigate to the GitDemo repository and create a .log file along with a log directory.

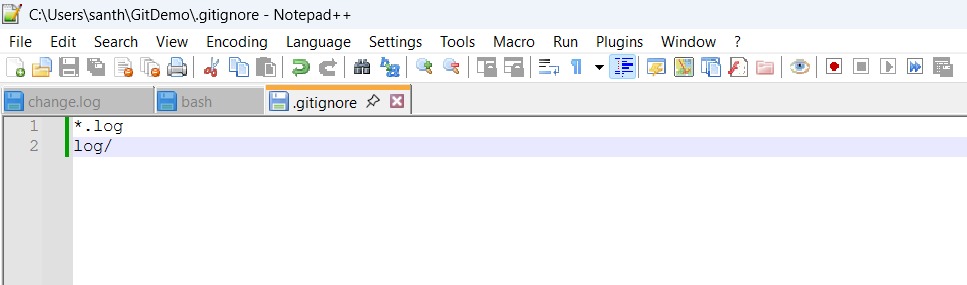


2.Create and modify the .gitignore file

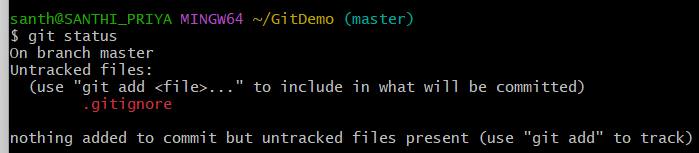


Add a rule to ignore all files with the .log extension.

Add a rule to ignore the entire log folder and its contents.



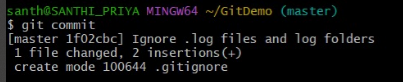
3.Check the current status of Git workspace using the git status command.

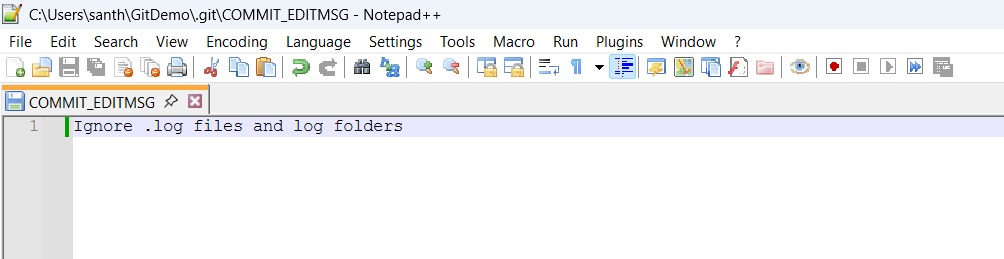


4.To include the .gitignore file in version tracking, use the Git add command.

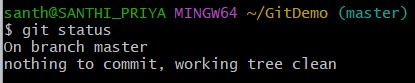


5.Save the changes by committing the updated .gitignore file.

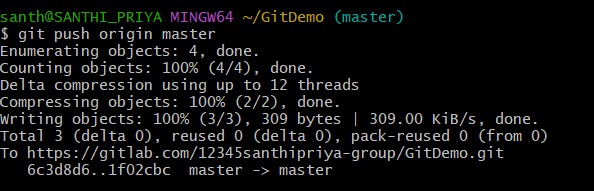


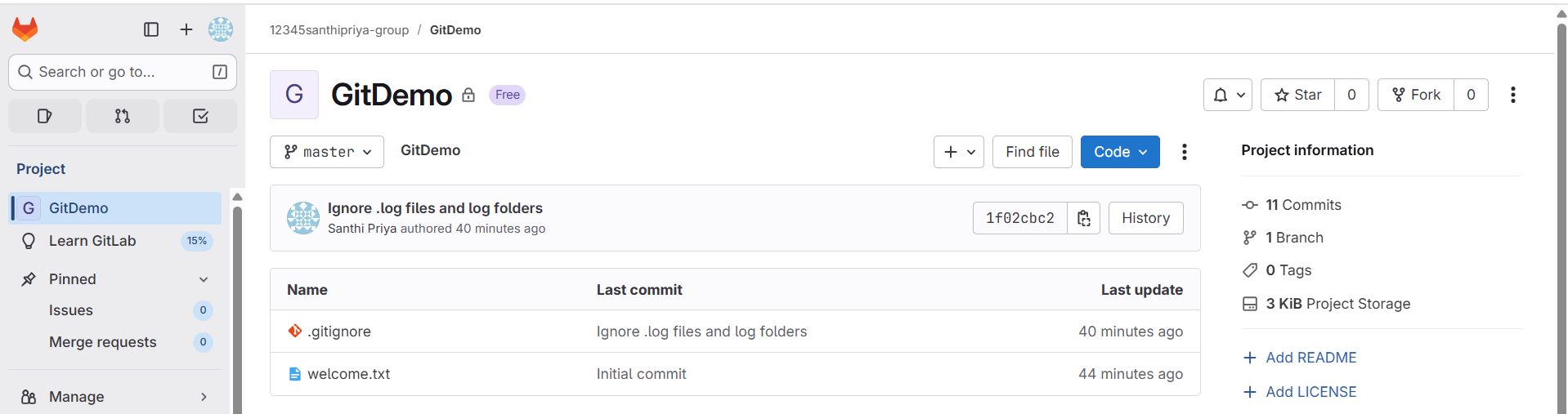


6.Run git status again to check the current status.



7.Push the commit to the remote repository.



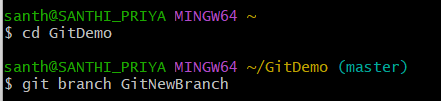




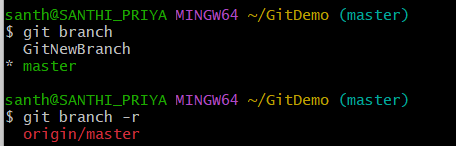
1. **GIT-HOL**

**Branching:**

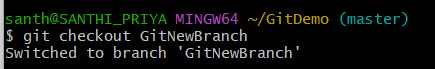
1.Create a new branch “GitNewBranch”.



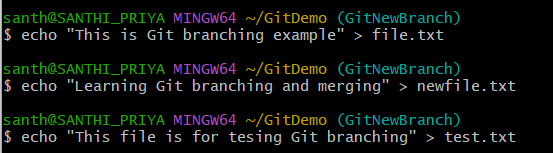
2. List all the local and remote branches available in the current trunk.



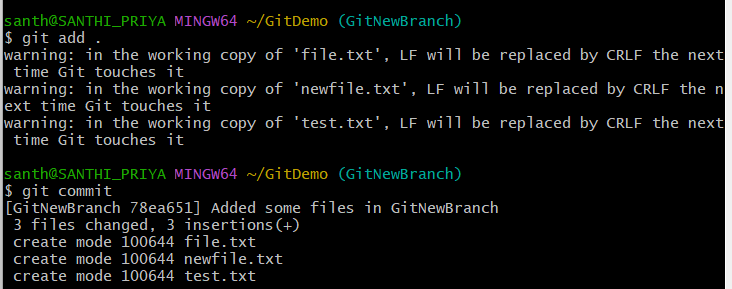
3.Switch to the newly created branch.

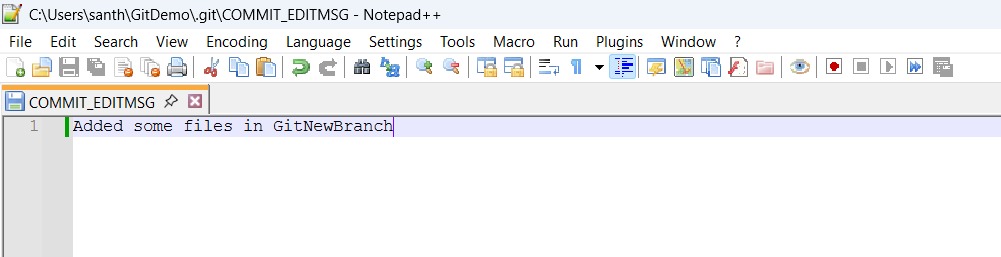


4.Add some files to it with some contents.

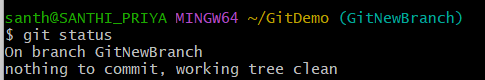


5.Commit the changes to the branch.





6.Check the status with “git status” command.

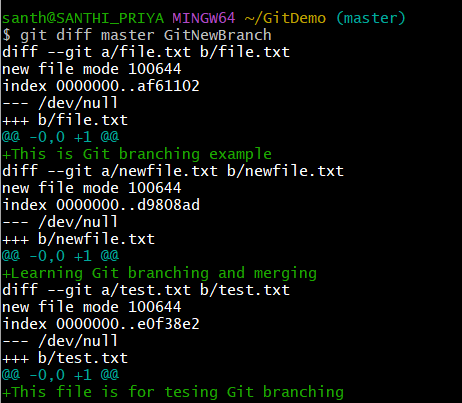
4

**Merging:**

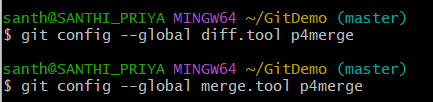
1.Switch to the master

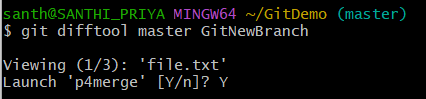


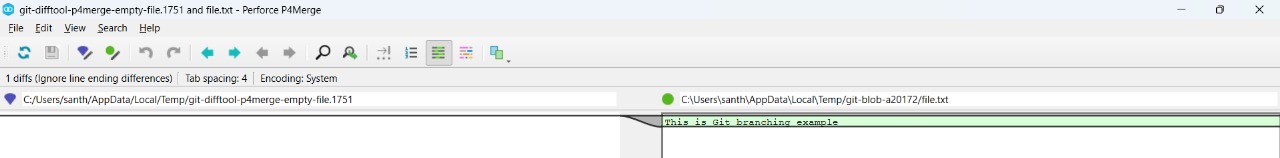
2. List out all the command-line differences between master and GitNewBranch.



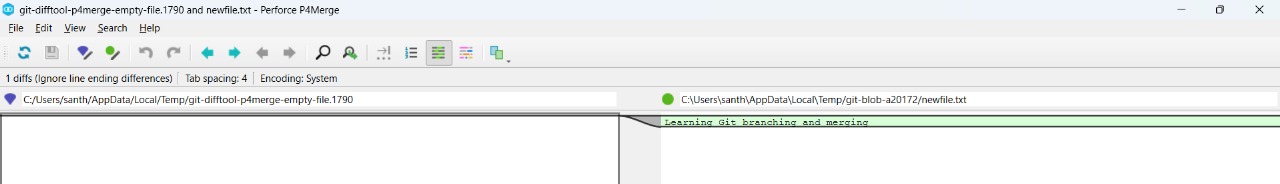
3.List out all the visual differences between master and branch using P4Merge tool.



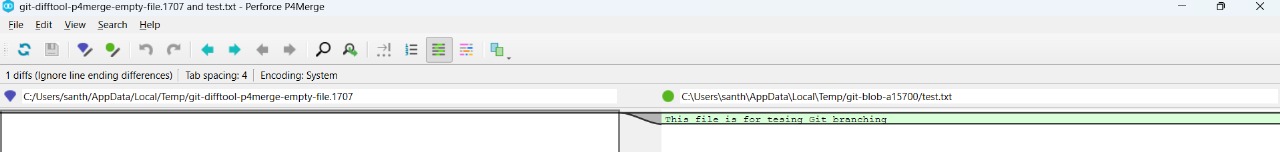




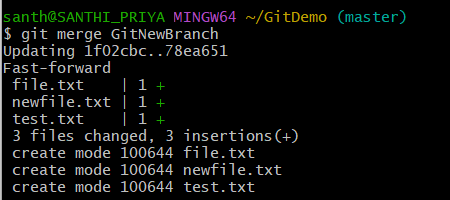




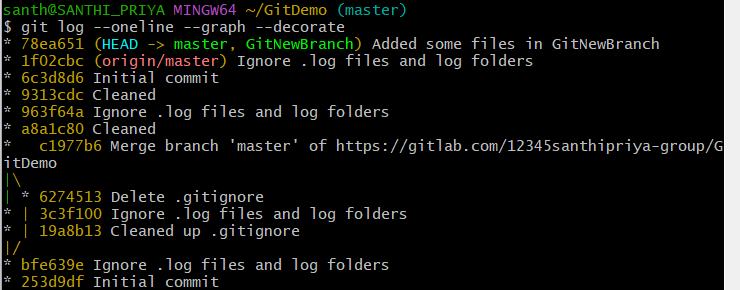




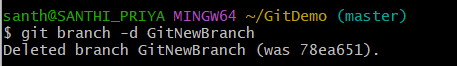
4.Merge the source branch to the trunk.



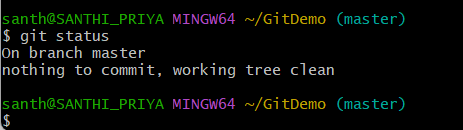
5.Observe the logging after merging using “git log –oneline –graph –decorate”



6. Delete the branch after merging with the trunk.

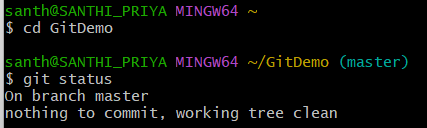


7.Observe the git status.

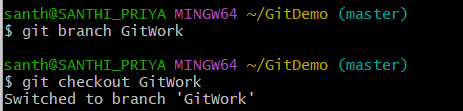


1. **GIT-HOL**

1.Verify if master is in clean state.



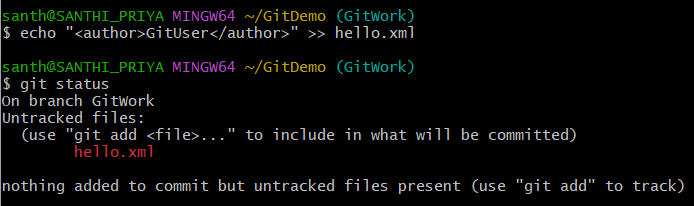
2.Create a branch “GitWork”.



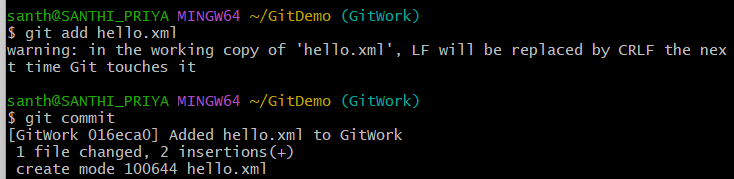
Add a file “hello.xml”.

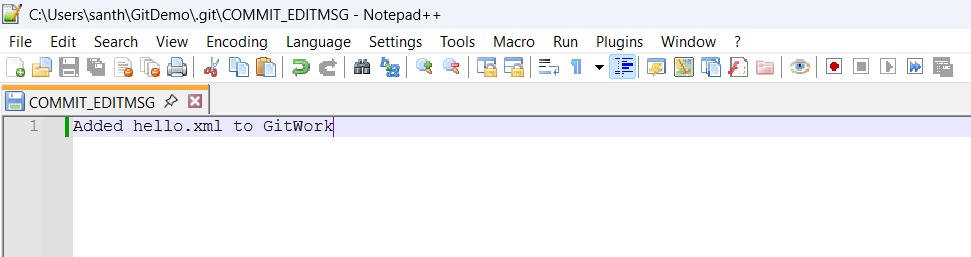


3.Update the content of “hello.xml” and observe the status.

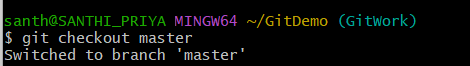


4.Commit the changes to reflect in the branch





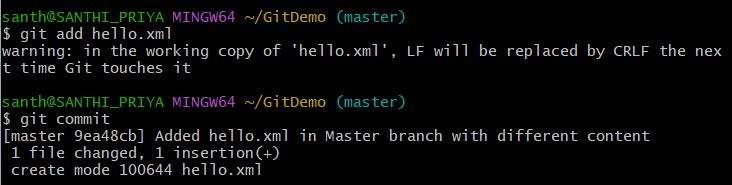
5.Switch to master.

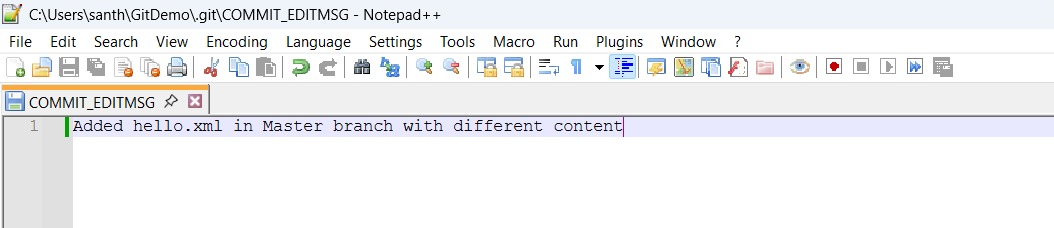


6.Add a file “hello.xml” to the master and add some different content than previous.

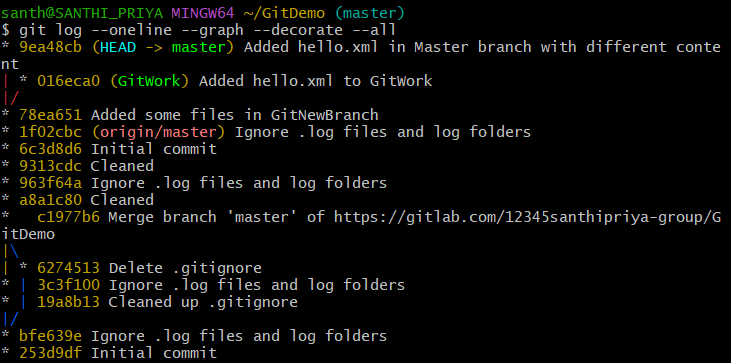


7.Commit the changes to the master.

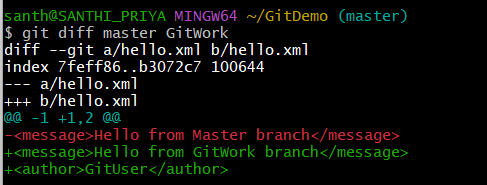




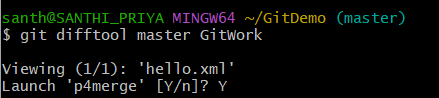
8.Observe the log by executing “git log –oneline –graph –decorate –all”

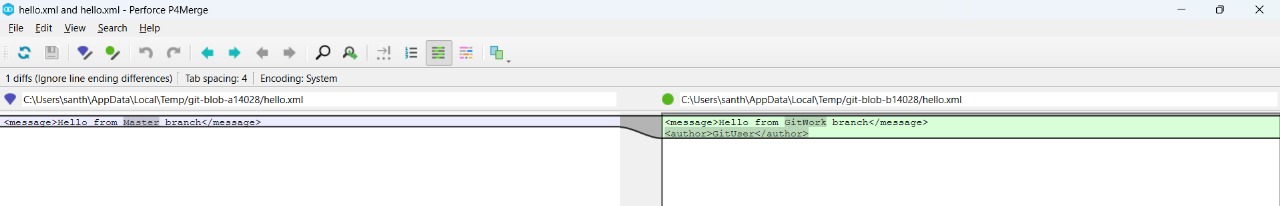


9.Check the differences with Git diff tool

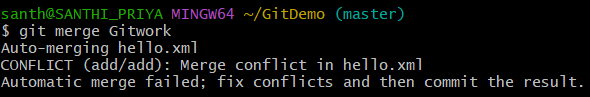


10.Use P4Merge tool to list out all the differences between master and branch

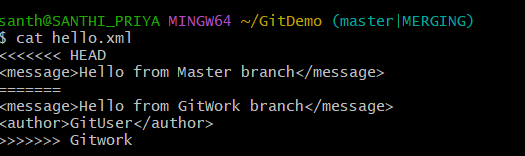




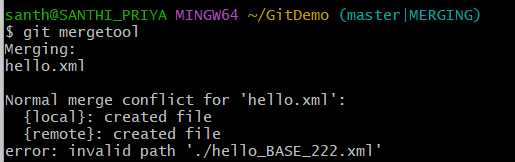
11.Merge the branch to the master.

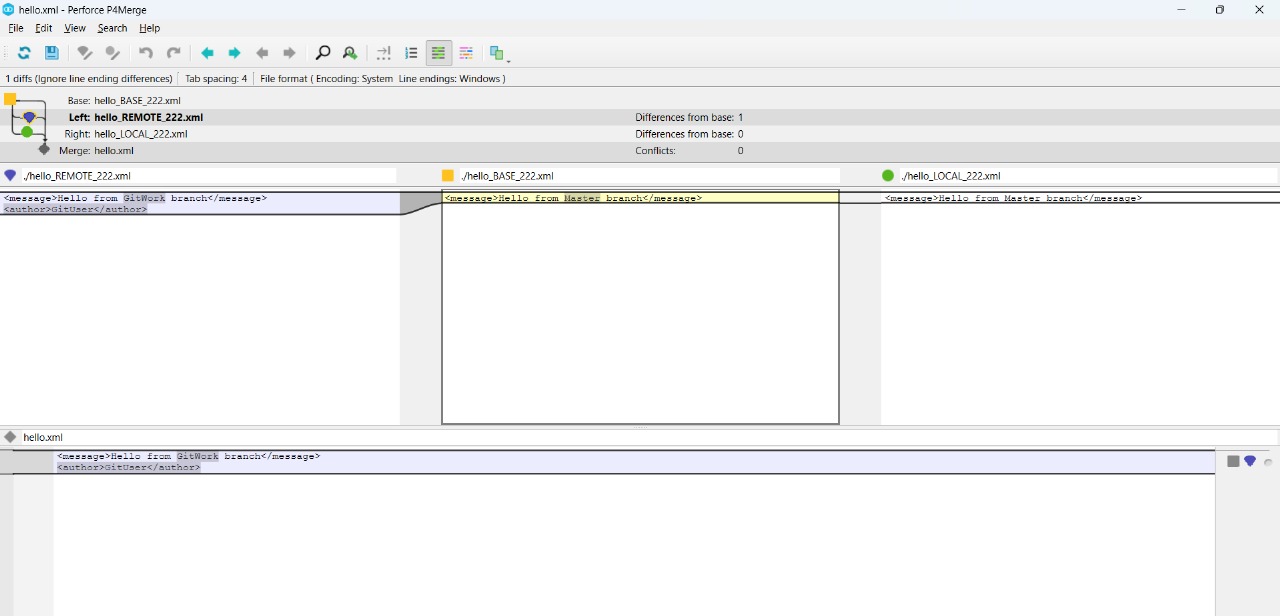
****

12.Observe the git mark up.

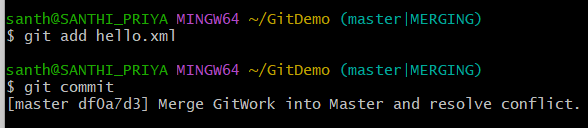


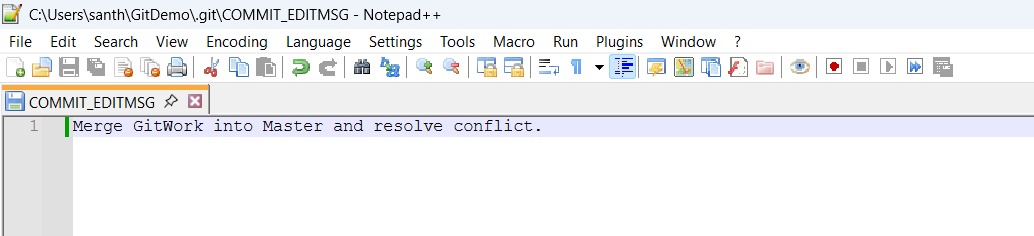
13.Use 3-way merge tool to resolve the conflict



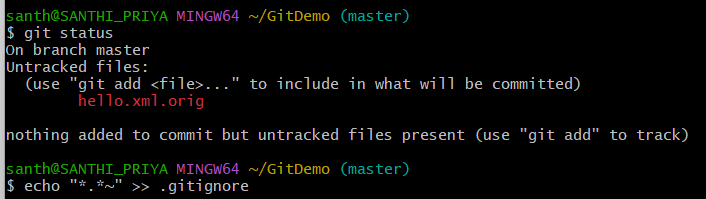


14.Commit the changes to the master, once done with conflict.

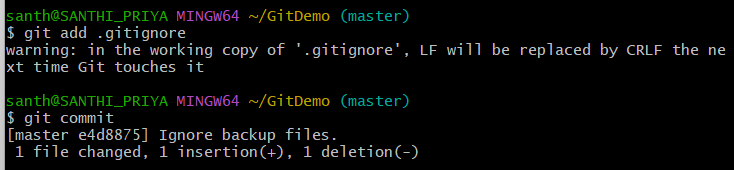


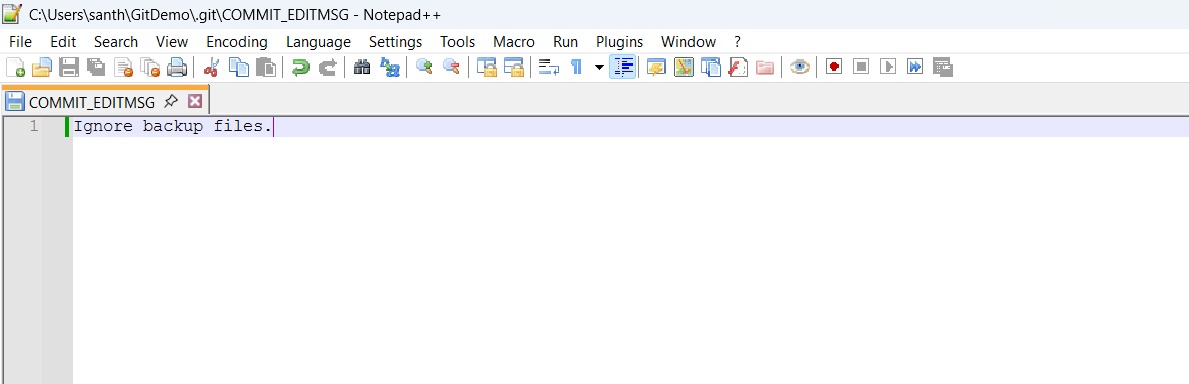


15.Observe the git status and add backup file to the .gitignore file.

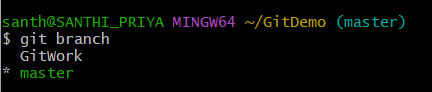


16.Commit the changes to the .gitignore.

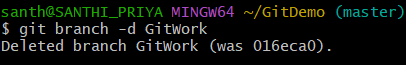




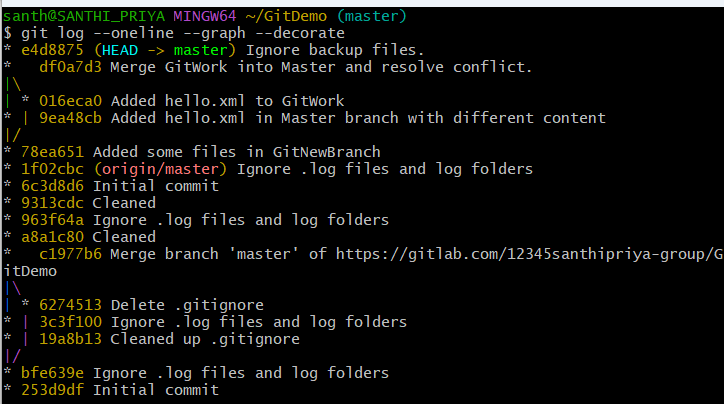
17.List out all the available branches.

****

18.Delete the branch, which merge to master.

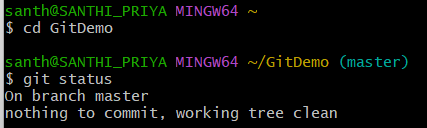


19.Observe the log by executing “git log –oneline –graph –decorate”.

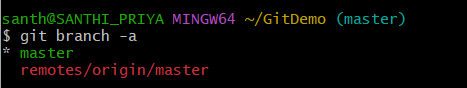


1. **GIT-HOL**

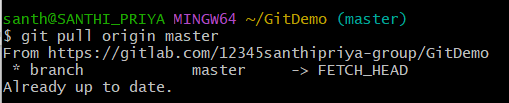
1.Verify if master is in clean state.



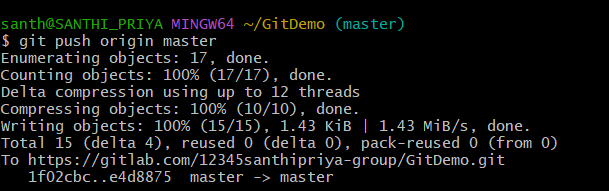
2.List out all the available branches.



3.Pull the remote git repository to the master.



4. Push the changes.



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

