# Git – Hands-on Lab Exercises

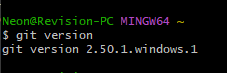
## Exercise 1

Create a local Git repository and configure username, email, and Notepad++ as default editor.

### Code:

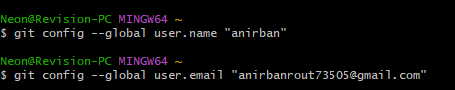
git –version

Expected Output:



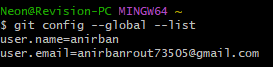
git config --global user.name "Your Name"  
git config --global user.email “[your@email.com](mailto:your@email.com)”

Expected Output:



git config –list

Expected Output:



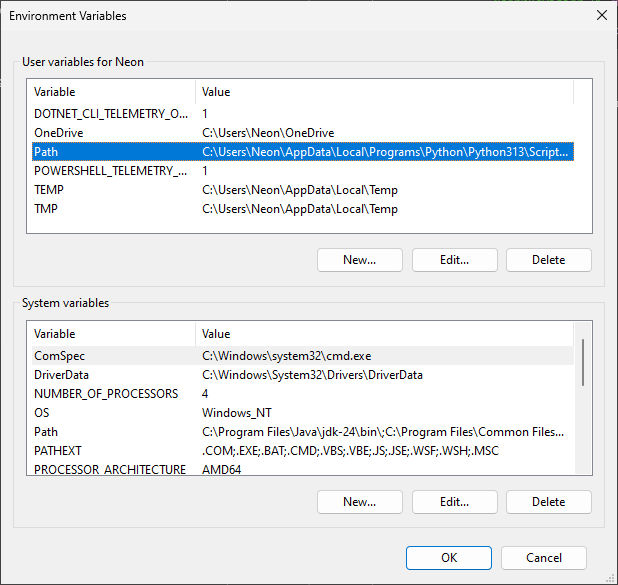
Step 2: Integrate notepad++.exe to git and make it a default editor

Expected Output:



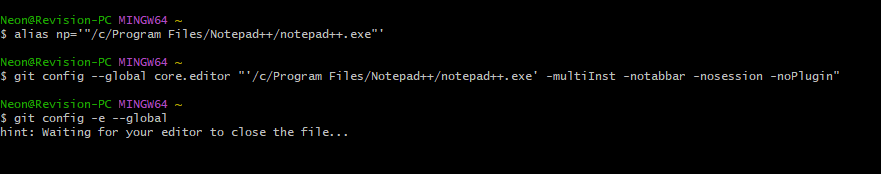
To add path of notepad++.exe to environment variable.

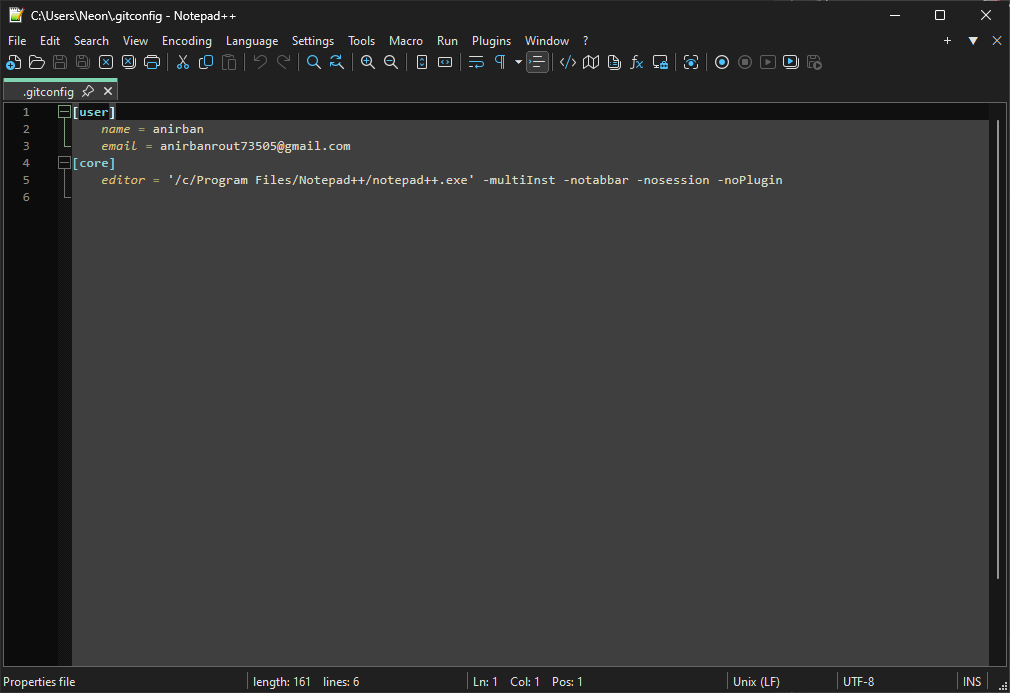
Expected Output:



alias np='"/c/Program Files/Notepad++/notepad++.exe"'  
git config --global core.editor "'/c/Program Files/Notepad++/notepad++.exe' -multiInst -notabbar -nosession -noPlugin"  
git config -e –global

Expected Output:

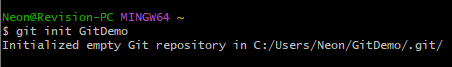




Step 3: Add a file to source code repository:

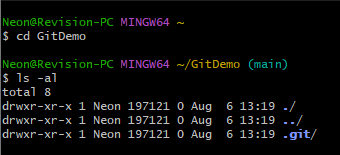
Creating a new project “GitDemo”

Expected Output:



To verify the initialization and display the file contents.

Expected Output:



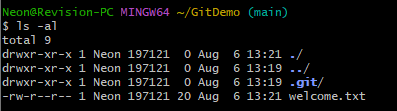
To create a new file “welcome.txt” and add content to the file.

Expected Output:



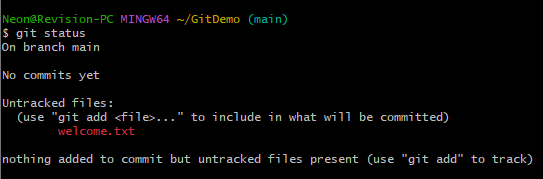
To verify if the file is properly add to the local git repo.

Expected Output:



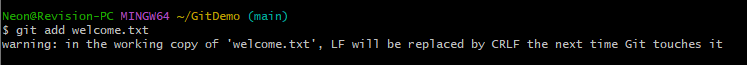
To check the status of the newly created file.

Expected Output:



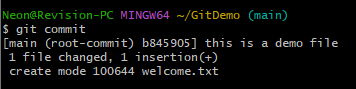
Add the file to the local repo.

Expected Output:



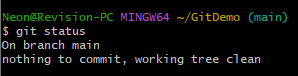
To make the file permanent in the local repo:

Expected Output:



Check the status of the local repo..

Expected Output:

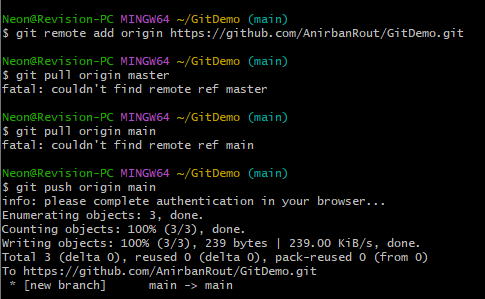


To create a remote or local repo of the global repo.

To pull remote repo.

To push local repo to global repo or main repo or github.

Expected Output:



## Exercise 2

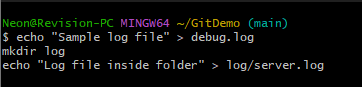
Create a “.log” file and a log folder in the working directory of Git. Update the .gitignore file in such a way that on committing these files (.log extensions and log folders) are ignored.

### Code:

Prepare Files for the repo:

echo "Sample log file" > debug.log  
mkdir log  
echo "Log file inside folder" > log/server.log

Expected Output:



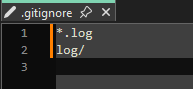
Create and Configure .gitignore:

Expected Output:



Add the following lines in notepad++:

Expected Output:



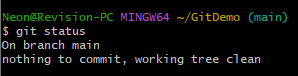
This tells git to ignore:

: all .log files

: the log folder and everything inside it

Finally check the git status:

Expected Output:



## Exercise 3

Create a new branch, make changes, and **merge it into master** using both command line (git diff) and visual diff tool (**P4Merge**). Understand branch creation, switching, and deletion.

### Code:

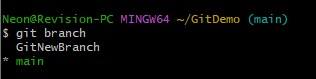
Create a new git branch “GitNewBranch”:

Expected Output:



List all the present branches of the repo:

Expected Output:



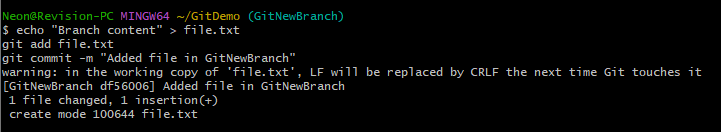
Switch to the newly created branch:

Expected Output:



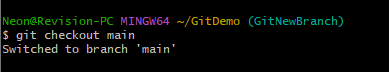
Create and add files to the newly created branch and add and commit them to the local repo:

Expected Output:



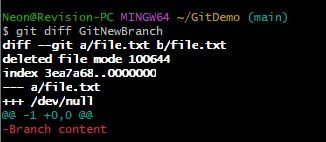
Switch to the main branch:

Expected Output:



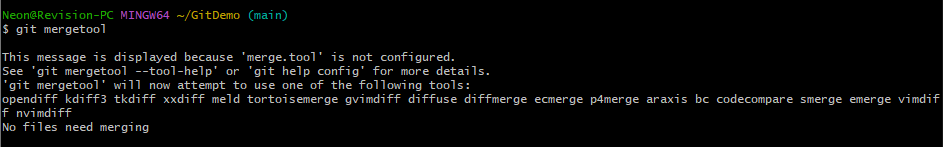
View CLI differences between the two branches:

Expected Output:



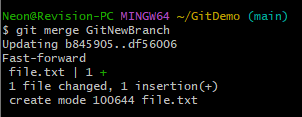
View visual differences using P4Merge:

Expected Output:



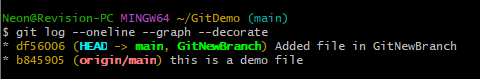
To merge the branch into main branch:

Expected Output:



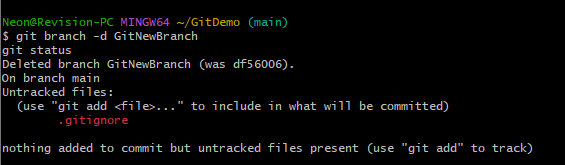
Now check for the merged log:

Expected Output:



Now delete the merges branch:

Expected Output:



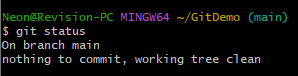
## Exercise 4

Simulate a **merge conflict** by editing the same file (hello.xml) in two branches, and resolve it using **3-way merge** and **P4Merge**.

### Code:

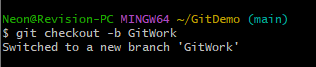
First verify the main branch is clean:

Expected Output:



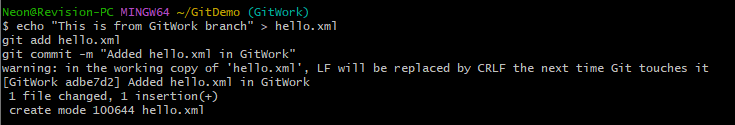
Now create a new branch and switch to it:

Expected Output:



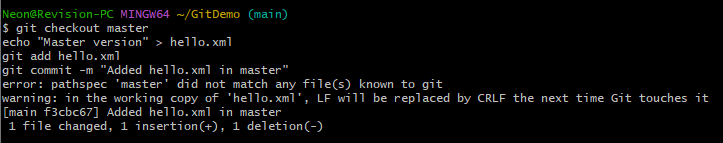
Add a file in the newly created branch:

Expected Output:



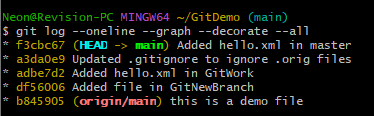
Now switch to the main branch and create conflicting file:

Expected Output:



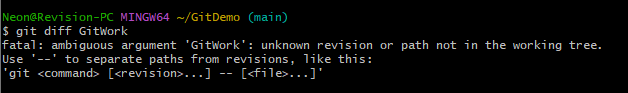
Now view logs visually:

Expected Output:



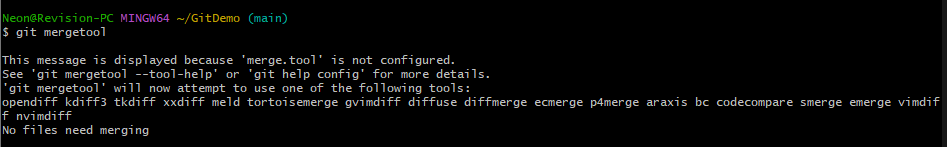
Check differences between the two branches:

Expected Output:



Launch visual merge tool (P4Merge):

Expected Output:



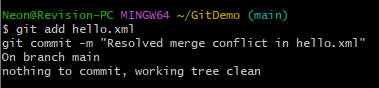
Now try to attempt merger and handle conflict:

Expected Output:



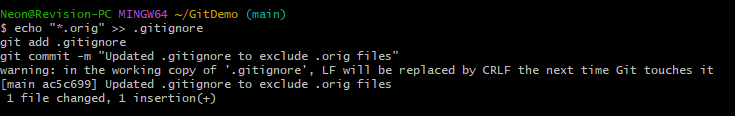
Resolve manually or using 3-way merge. Then …

Expected Output:



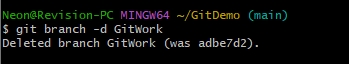
Now Ignore the backup files:

Expected Output:



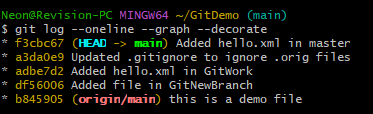
Clean up merged branch:

Expected Output:



After that do the final log check:

Expected Output:



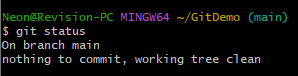
## Exercise 5

Clean up your Git workspace and **push changes** from local to **remote repository** using git pull and git push.

### Code:

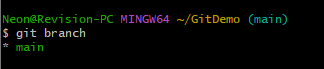
Verify to check if the master branch is clean:

Expected Output:



Now list all the branches of the repo:

Expected Output:



After that pull the latest changes from the remote repo and push the local changes to the remote repo or the github repo:

Expected Output:

