**Week 8 Git HandsOn Solutions**

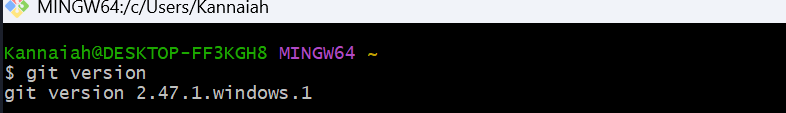
**Exercise -1**

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

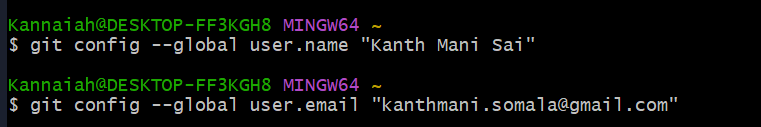
To create a new repository, signup with GitLab and register your credentials

Login to GitLab and create a “GitDemo” project

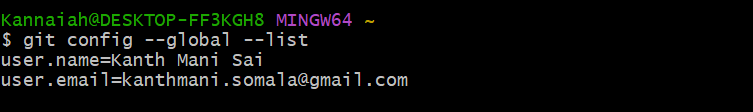
1. To check if Git client is installed properly: Open Git bash shell and execute



1. To configure user level configuration of user ID and email ID execute



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

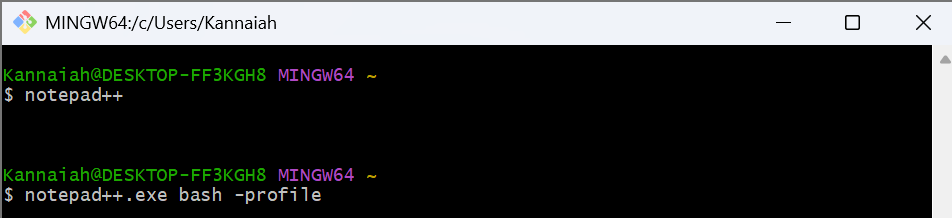


**Step 2: Integrate notepad++.exe to Git and make it a default editor**

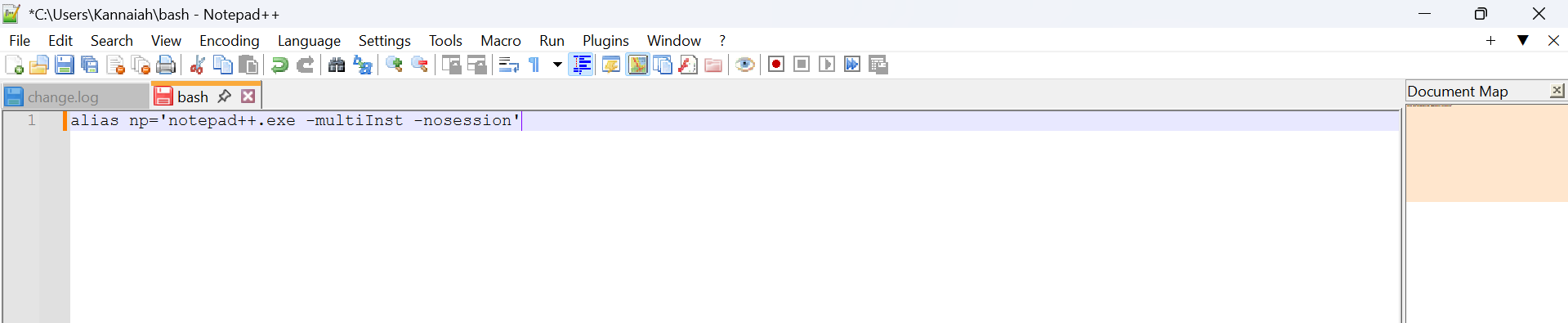
1. To check, if notepad++.exe execute from Git bash
2. Exit Git bash shell, open bash shell and execute

Now, notepad++ will open from Git bash shell

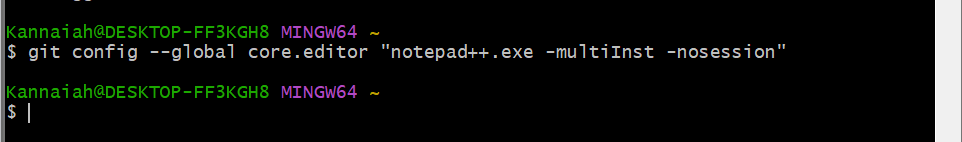
1. To create an alias command for notepad++.exe, execute



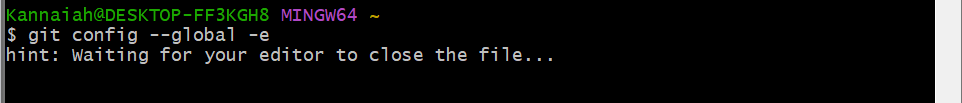
It will open notepad++ from bash shell, and create a user profile by adding the line in notepad++



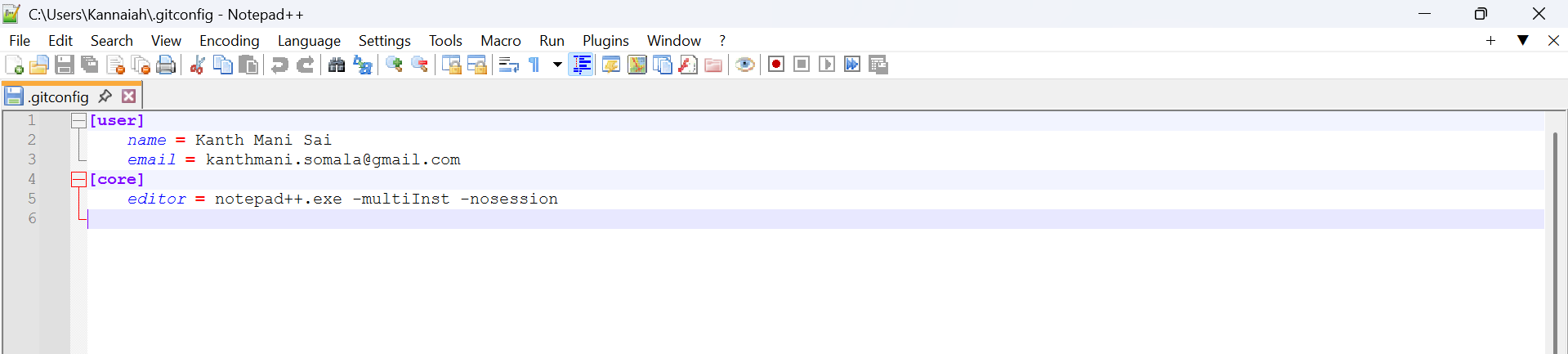
1. To configure the editor, execute the command



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

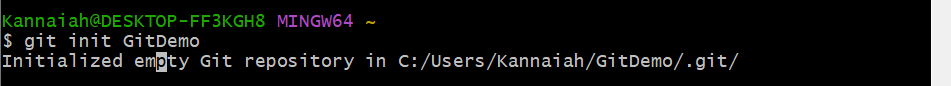
****

It will show the entire global configuration as shown below,

****

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

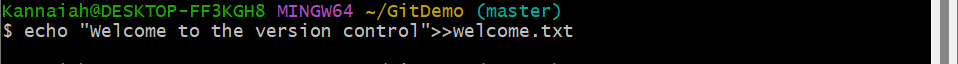
1. Open Git bash shell and create a new project “GitDemo” by executing the command



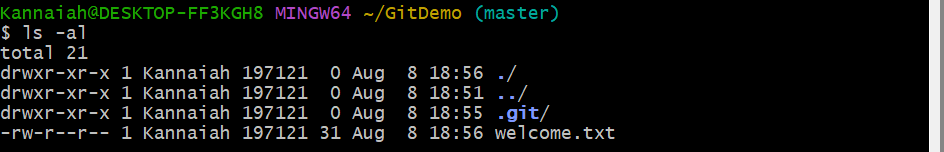
1. Git bash initializes the “**GitDemo**” repository. To verify, execute the command



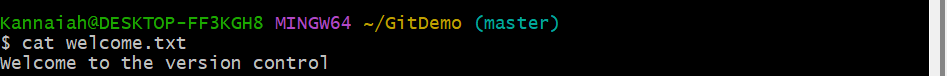
1. To create a file **“welcome.txt”** and add content to the file, execute the command

****

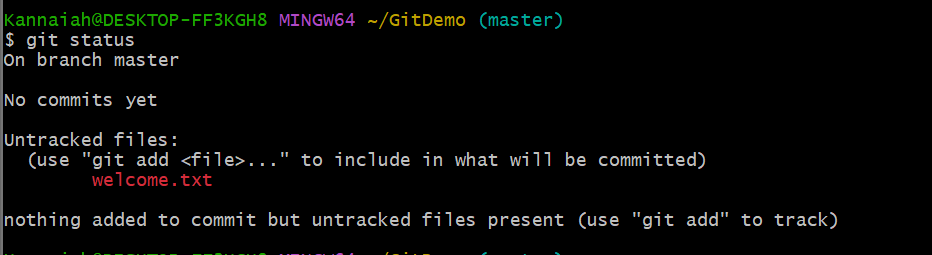
1. To verify if the file “welcome.txt” is created, execute

****

1. To verify the content, execute the command

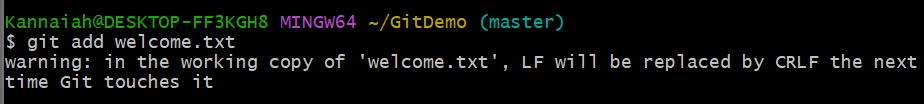


1. Check the status by executing

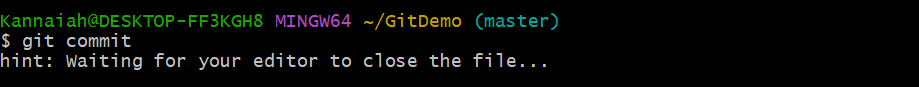


Now the file **“welcome.txt”** is available in Git “working directory”

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

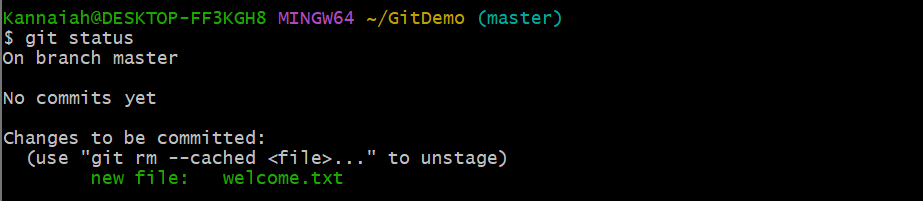


1. To add multi line comments, we are opening default editor to comment. Execute the command



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

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



**welcome.txt** is added to the local repository.

1. Signup with GitLab and create a remote repository **“GitDemo”**
2. To pull the remote repository, execute



1. To push the local to remote repository, execute



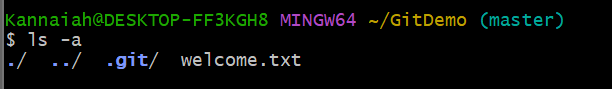
**Exercise-2**

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.

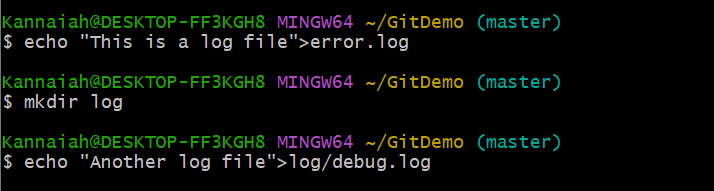
Verify if the git status reflects the same about working directory, local repository and git repository.

Steps:

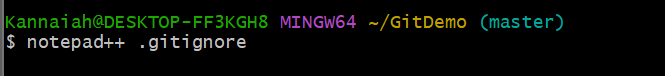
1. Go to your working directory



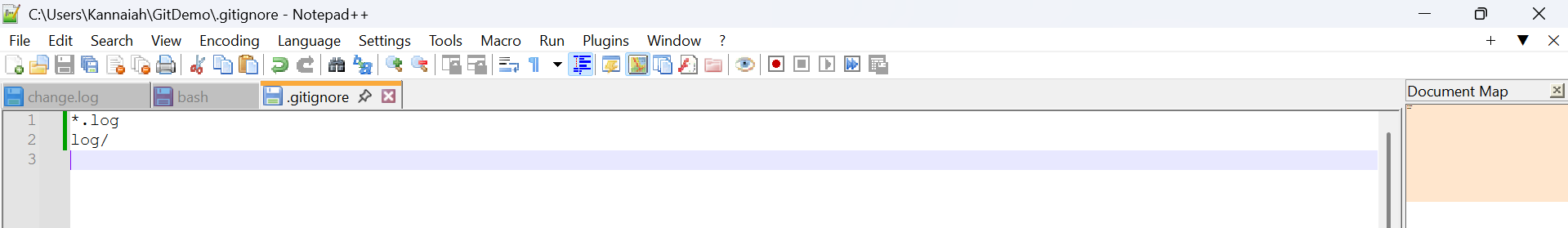
1. Create a .log file and log folder



1. Create .gitignore file



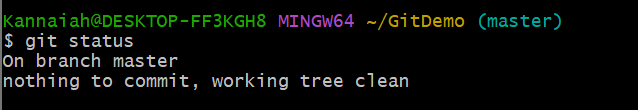
Add these lines:



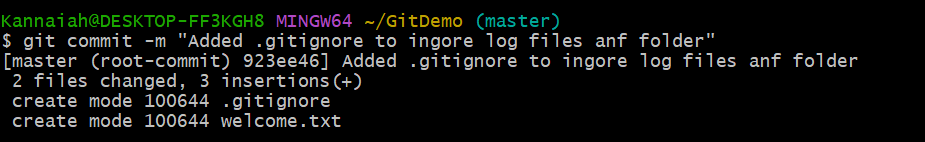
\*.log → ignores all files ending with .log.

log/ → ignores the whole log folder.

1. Check Git status



1. Commit .gitignore file



**Exercise-3**

**Branching:**

1. Create a new branch **“GitNewBranch”.**
2. List all the local and remote branches available in the current trunk. Observe the “\*” mark which denote the current pointing branch.
3. Switch to the newly created branch. Add some files to it with some contents.
4. Commit the changes to the branch.
5. Check the status with **“git status”** command.

**Merging:**

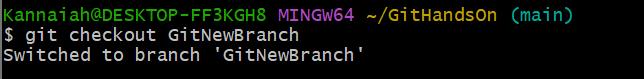
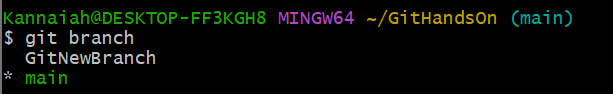
1. Switch to the master
2. List out all the differences between trunk and branch. These provide the differences in command line interface.
3. List out all the visual differences between master and branch using **P4Merge tool**.
4. Merge the source branch to the trunk.
5. Observe the logging after merging using **“git log –oneline –graph –decorate”**
6. Delete the branch after merging with the trunk and observe the git status.

Steps:

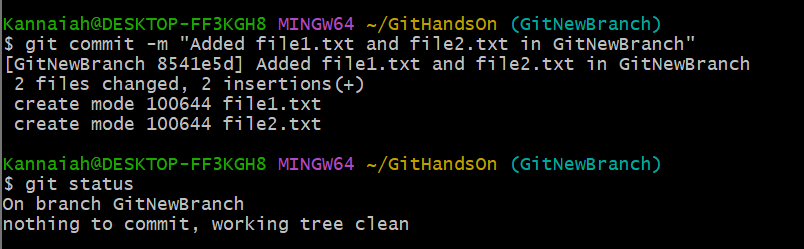
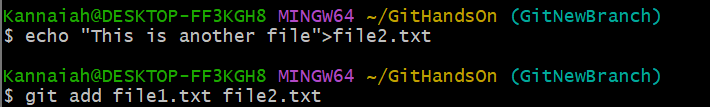
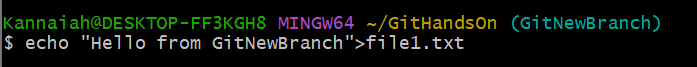
1.Clone the Repository to Your PC



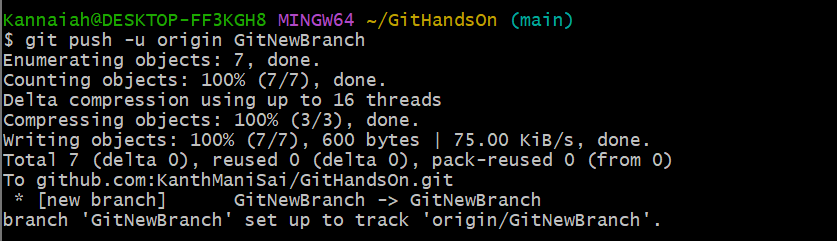
2. Create a New Branch



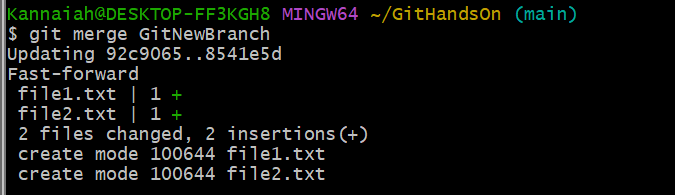
3. Add Files and Commit

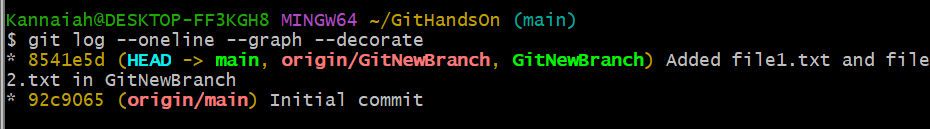


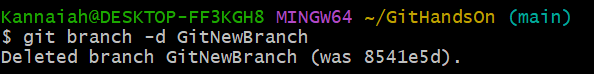
4.Push the branch to GitHub



5.Merge and Delete Branch





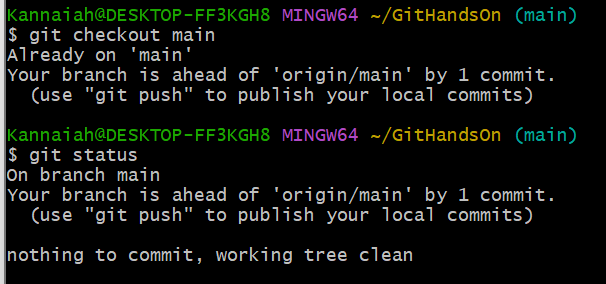


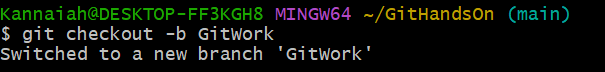


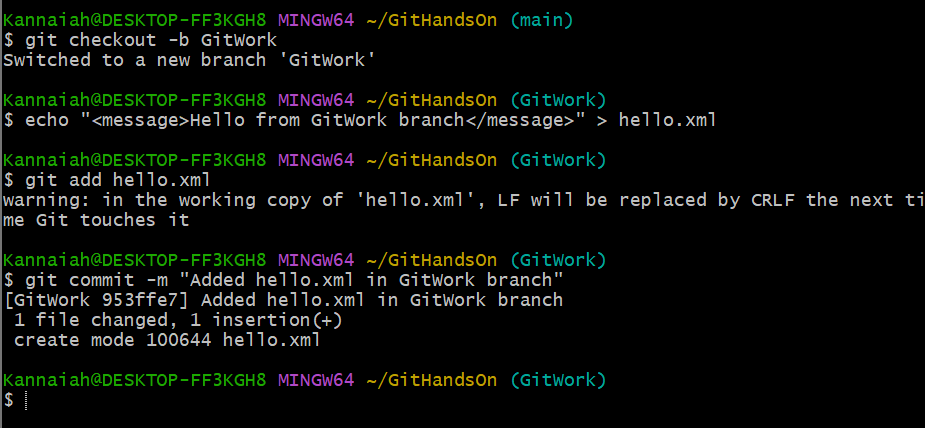
**Exercise-4:**

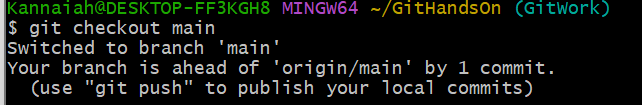
Steps:

1. Verify if master is in clean state.

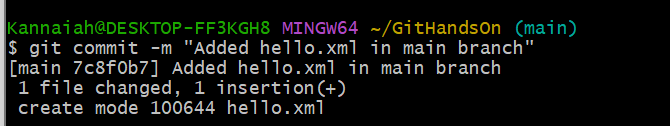


1. Create a branch **“GitWork”.** Add a file “hello.xml”. 
2. Update the content of “hello.xml” and observe the status
3. Commit the changes to reflect in the branch

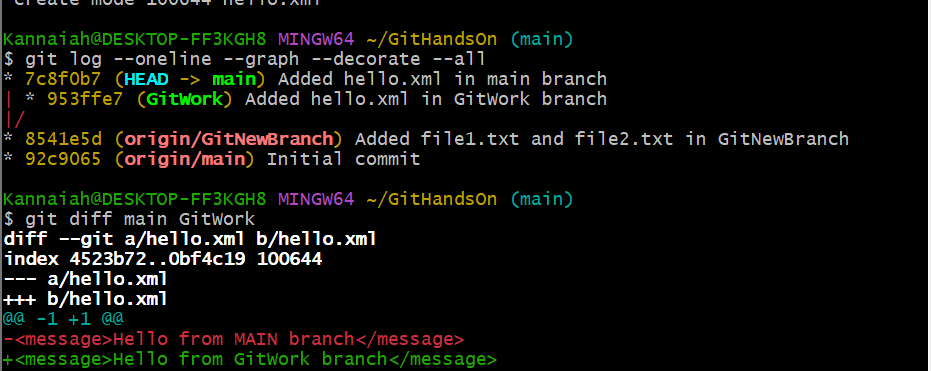


1. Switch to master. 
2. Add a file **“hello.xml”** to the master and add some different content than previous. 
3. Commit the changes to the master

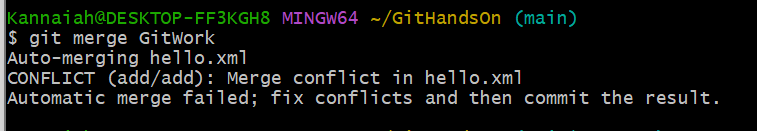


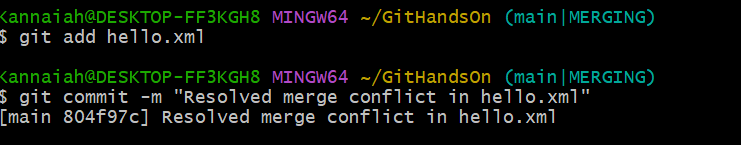


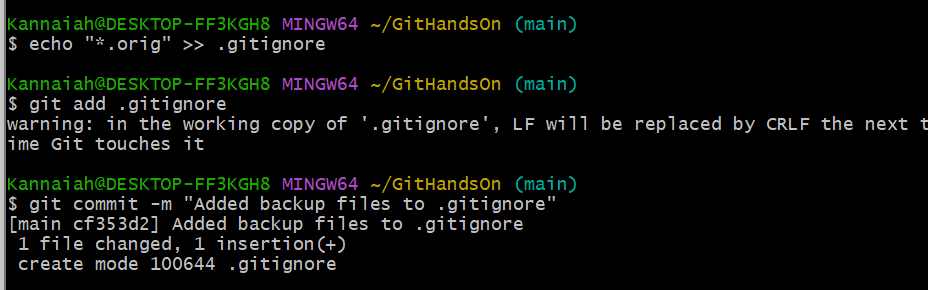
1. Observe the log by executing **“git log –oneline –graph –decorate –all”**
2. Check the differences with Git diff tool
3. For better visualization, use P4Merge tool to list out all the differences between master and branch

****

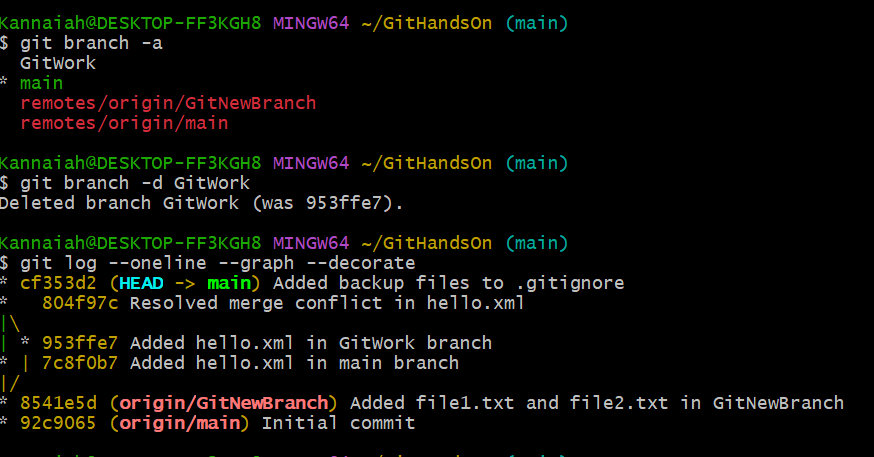
1. Merge the bran to the master

****

1. Observe the git mark up.
2. Use 3-way merge tool to resolve the conflict
3. Commit the changes to the master, once done with conflict 
4. Commit the changes to the .gitignore



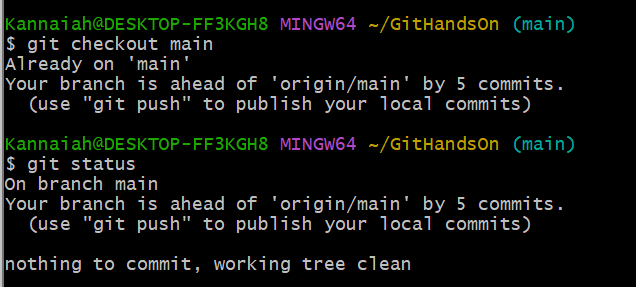
1. List out all the available branches
2. Delete the branch, which merge to master.
3. Observe the log by executing **“git log –oneline –graph –decorate”**



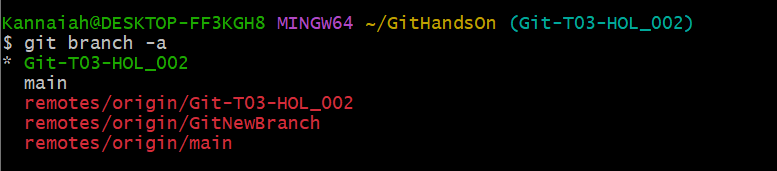
**Exercise-5**

Steps:

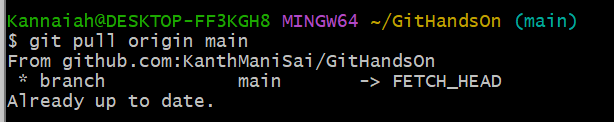
1. Verify if master is in clean state.



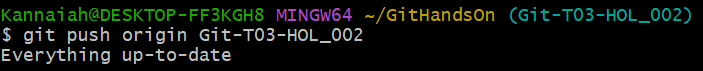
1. List out all the available branches.



1. Pull the remote git repository to the master



1. Push the changes, which are pending from **“Git-T03-HOL\_002”** to the remote repository.



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

