**HANDS ON -1 (filename:- 1.Git – HOL)**

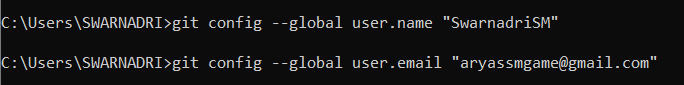
**Step 1: Setup your machine with Git Configuration  
To create a new repository, signup with GitLab and register your credentials  
Login to GitLab and create a “GitDemo” project  
1. To check if Git client is installed properly: Open Git bash shell and execute  
If output shows Git with its version information that indicates, that Git Client installs  
properly.**

**=**

****

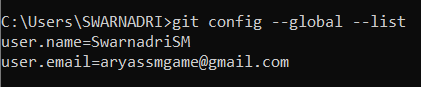
**2. To configure user level configuration of user ID and email ID execute**

**=**

****

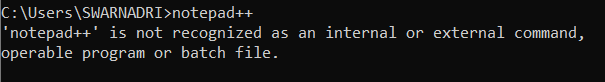
**3. To check if the configuration is properly set, execute the following command.**

**=**

****

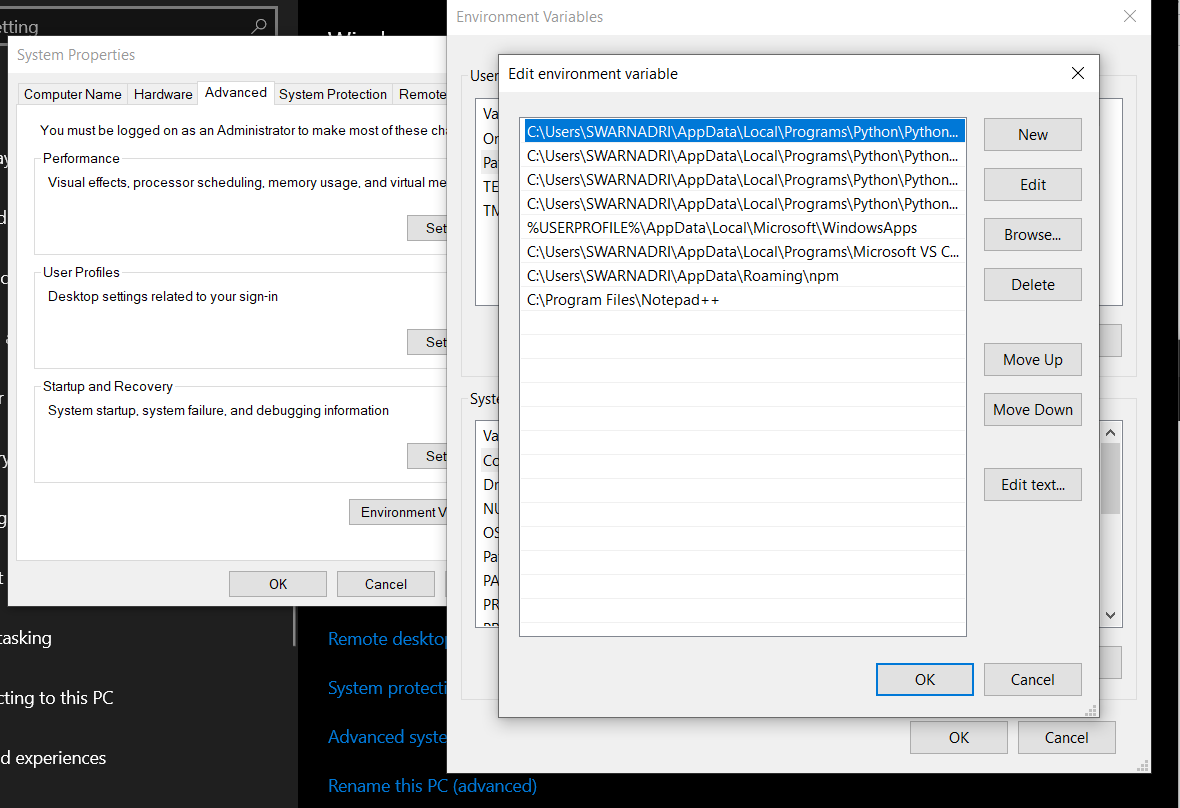
**Step 2: Integrate notepad++.exe to Git and make it a default editor  
1. To check, if notepad++.exe execute from Git bash  
If Git bash could not able to recognize notepad++ command that implies notepad++.exe is note added to the environment path variable.**

**=**

****

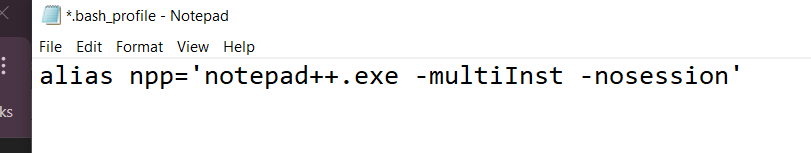
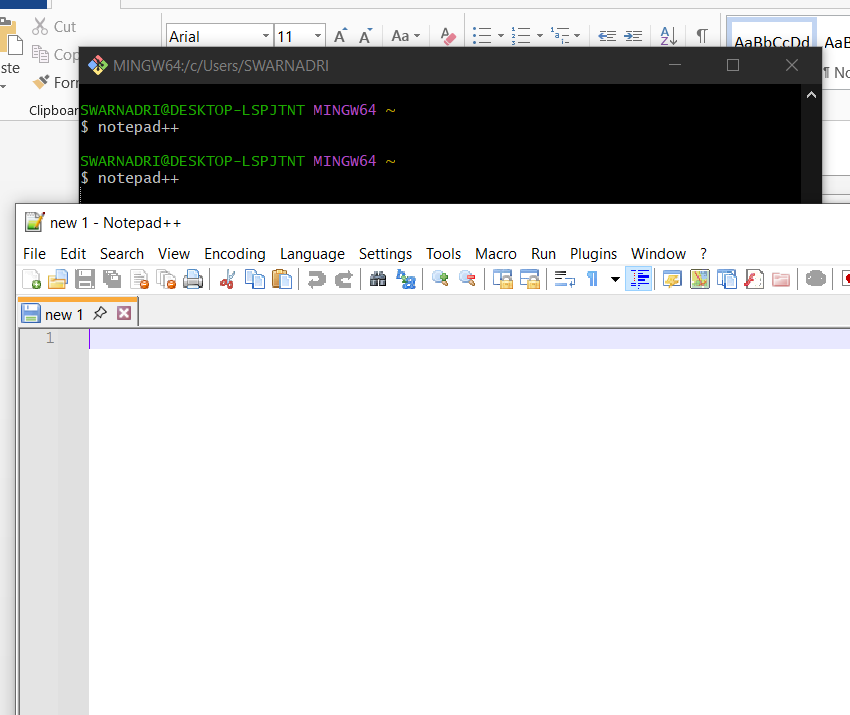
**To add path of notepad++.exe to environment variable, go to control panel -> System -> Advanced System settings. Go to Advanced tab -> Environment variables -> Add path of notepad++.exe to the path user variable by clicking on “Edit”**

**=**

**=**

**2. Exit Git bash shell, open bash shell and execute  
Now, notepad++ will open from Git bash shell**

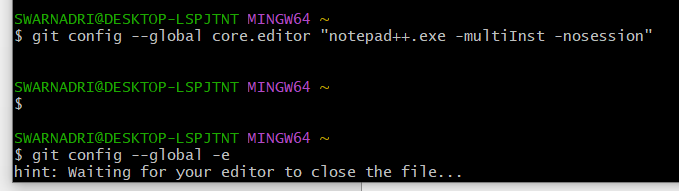
**=**

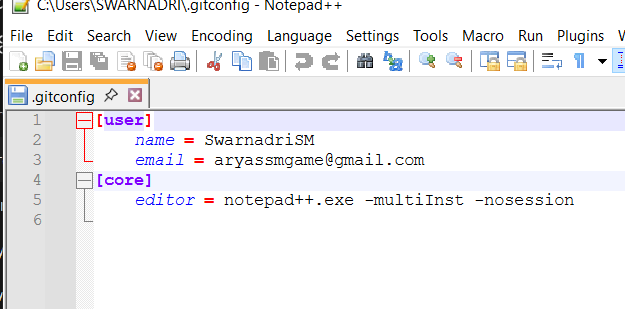
**  
3. To create an alias command for notepad++.exe, execute  
It will open notepad++ from bash shell, and create a user profile by adding the line in  
notepad++**

=

4. To configure the editor, execute the command git config --global core.editor "notepad++.exe -multiInst -nosession"  
5. To verify if notepad++ is the default editor, execute the command git config --global -e  
Here ‘-e’ option implies editor

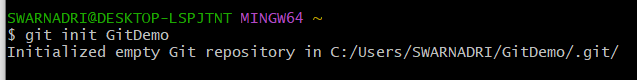
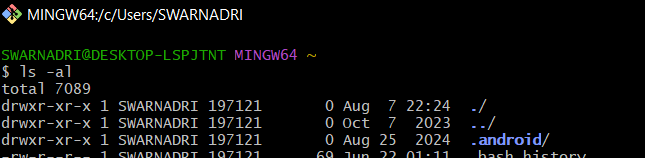
=





**Step 3: Add a file to source code repository  
1. Open Git bash shell and create a new project “GitDemo” by executing the command**

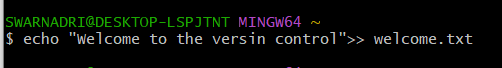
**=**

**  
2. Git bash initializes the “GitDemo” repository. To verify, execute the command  
It will display all the hidden files in the Git “working directory”.**

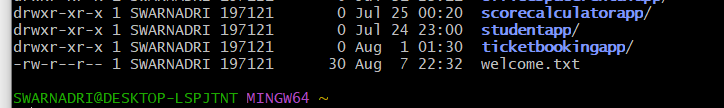
**=**

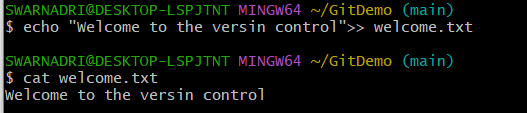
**3. To create a file “welcome.txt” and add content to the file, execute the command**

**-=**

****

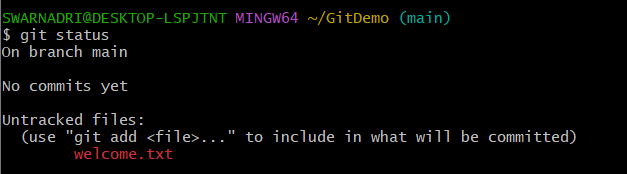
**4. To verify if the file “welcome.txt” is created, execute ls -al**

=

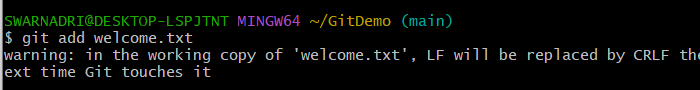
**5. To verify the content, execute the command**

**=**

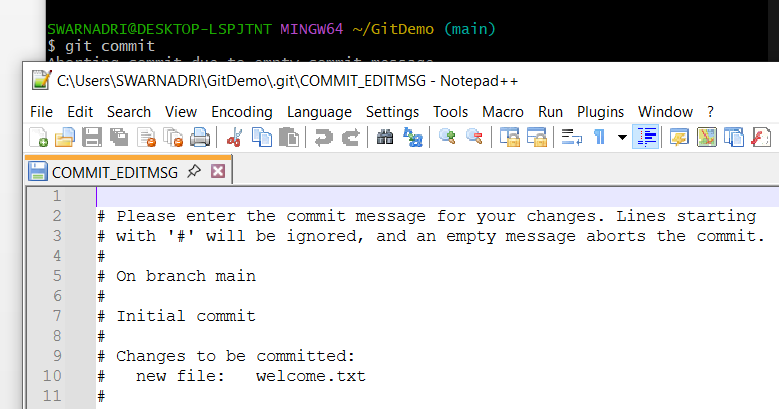
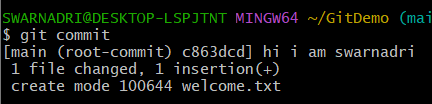
**6. Check the status by executing**

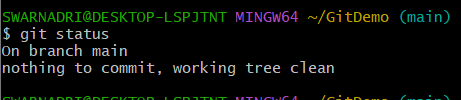
**=**

**Now the file “welcome.txt” is available in Git “working directory”  
7. To make the file to be tracked by Git repository, execute the command**

**= **

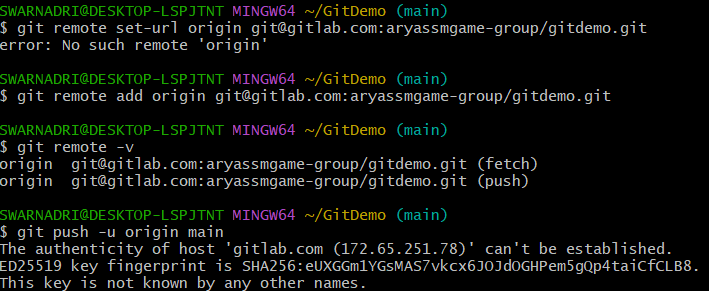
**8. To add multi line comments, we are opening default editor to comment. Execute the  
command  
=**

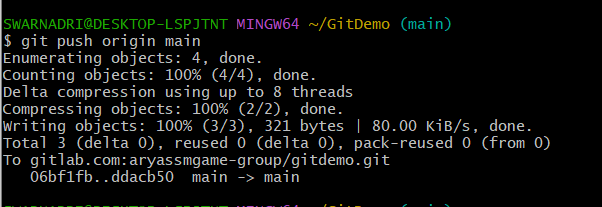
**   
9. To check if local and “Working Directory” git repository are same, execute git status  
welcome.txt is added to the local repository.**

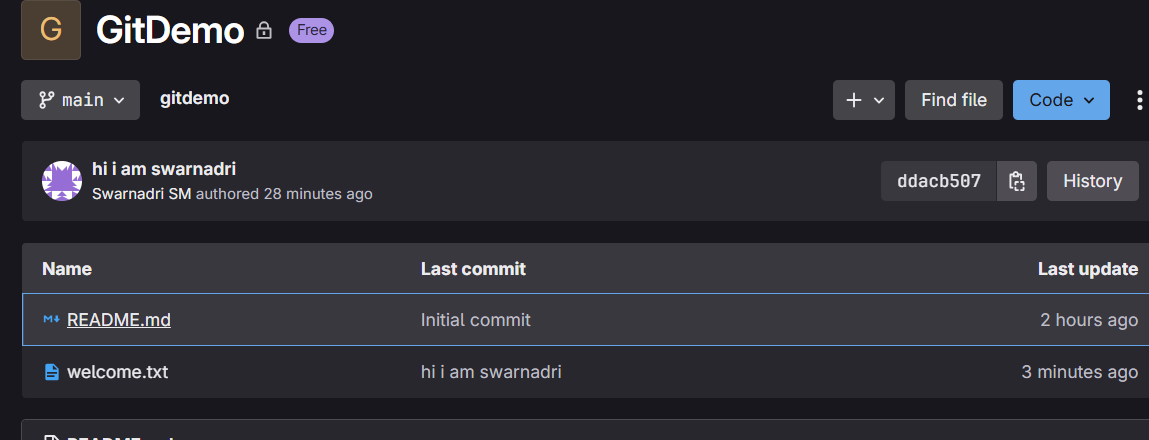
**= **

**10. Signup with GitLab and create a remote repository “GitDemo”  
11. To pull the remote repository, execute  
git pull origin master  
12. To push the local to remote repository, execute  
git push origin master**

**=**

****





WE CAN SEE THE WELCOME.TXT IS PUSHED

**HANDS ON – 2 ( file name: 2.Git – HOL)**

**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.**

**=** log file created





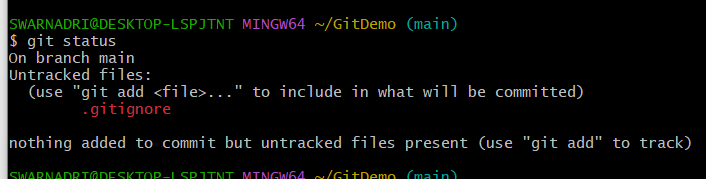
Log directory made

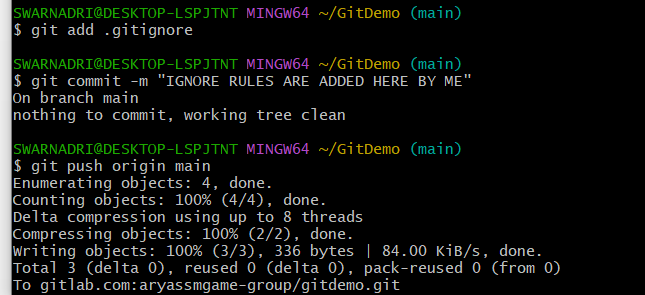


Example\_SSM log file created inside that folder

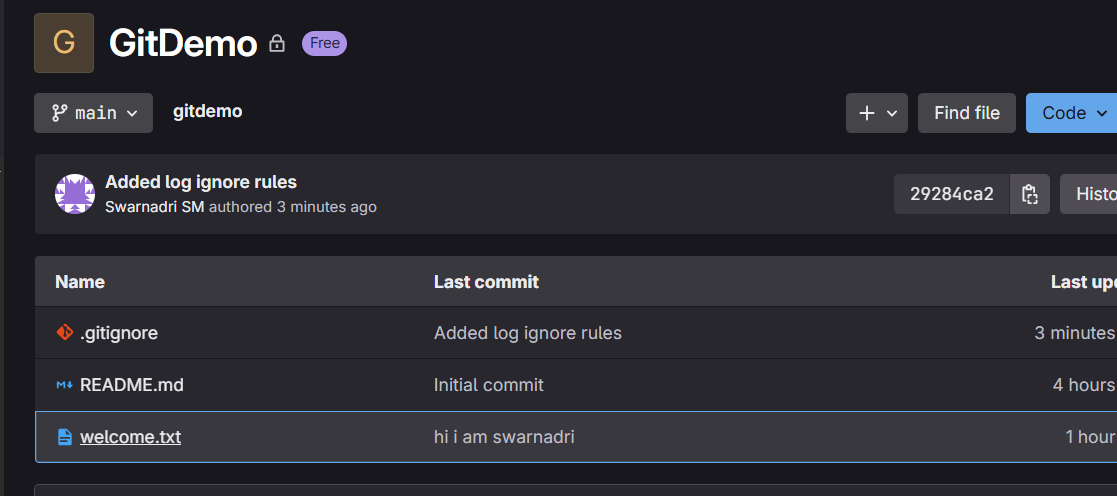
**Verify if the git status reflects the same about working directory, local repository and git repository**

**=**



Next added the gitignoore and commit is done

RESULT IN GITLAB:



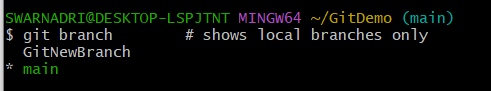
**HANDS ON – 3 ( file name: 3.Git – HOL)**

**Branching:  
1. Create a new branch “GitNewBranch”.**

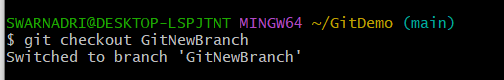
**=**

**2. List all the local and remote branches available in the current trunk. Observe the “\*” mark which denote the current pointing branch.**

**=**

****

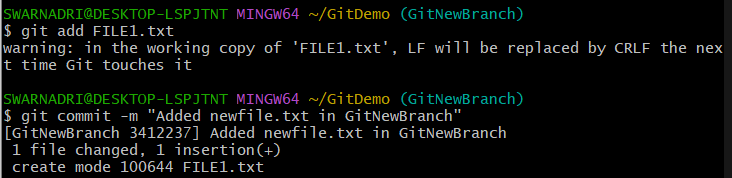
**3. Switch to the newly created branch. Add some files to it with some contents.**

**=**

**4. Commit the changes to the branch.**

**=**

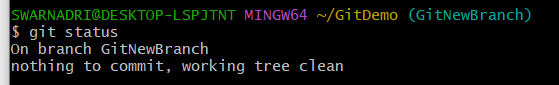
****

****

I creted a new file and added this in the new brnch and also made a commit

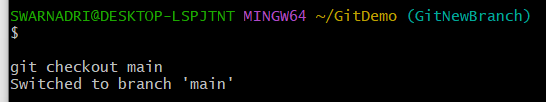
**5. Check the status with “git status” command.**

**=**

****

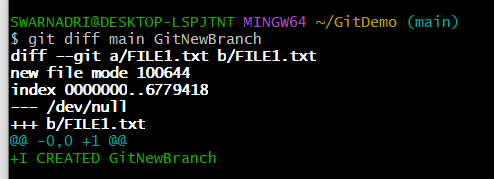
**Merging:  
1. Switch to the master ( I HAVE USED MAIN AS NAMING CONVENTION)**

**=**

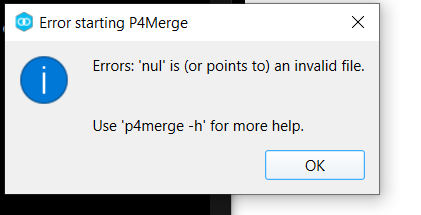
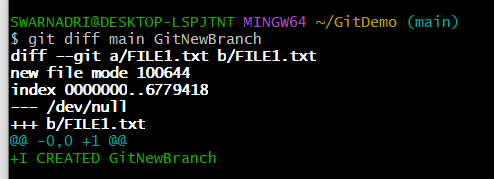
****

**2. List out all the differences between trunk and branch. These provide the differences in command line interface.**

**=**

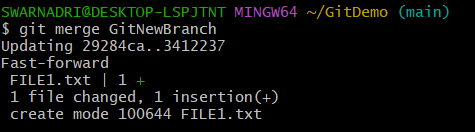
****

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

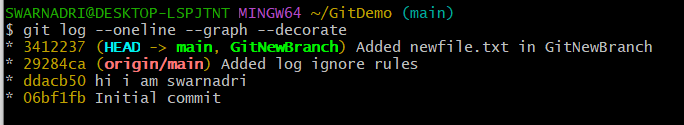
**=**

P4MERGE IS CONSTANTLY GIVING ERRORS IN MY LCAL MACHINE, SO I USED THE git diff main GitNewBranch -- FILE1.txt COMMAND

**4. Merge the source branch to the trunk.**

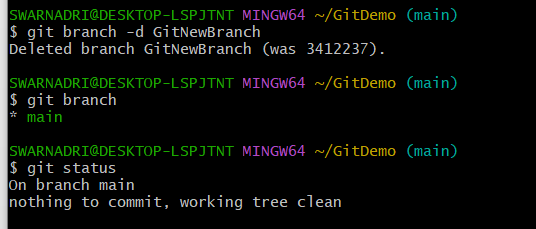
**=**

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

**=**

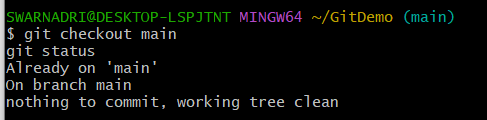
**6. Delete the branch after merging with the trunk and observe the git status**.

=

****

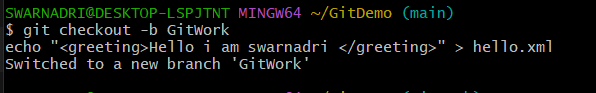
**HANDS ON – 4 ( file name: 4.Git – HOL)**

**Please follow the instructions to complete the hands-on. Each instruction expect a command for the Git Bash.**

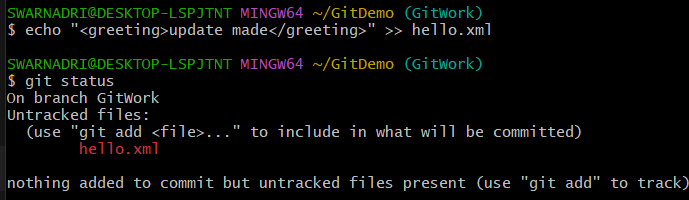
1. **Verify if main is in clean state.**

**=**

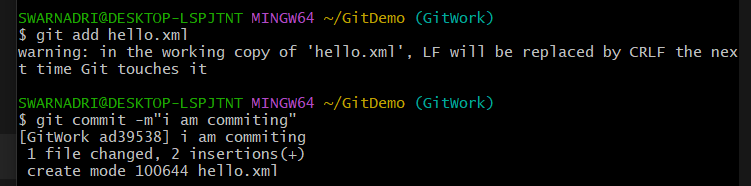
1. **Create a branch “GitWork”. Add a file “hello.xml”.**

**=**

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

**=**

1. **Commit the changes to reflect in the branch**

**=**

1. **Switch to main.**

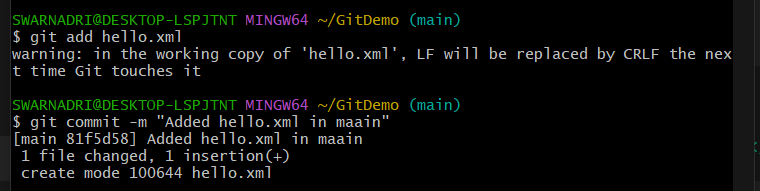
**=**

****

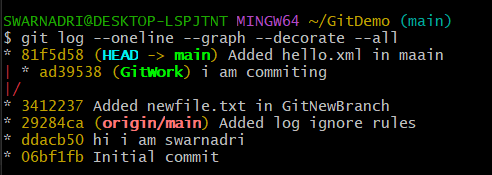
1. **Add a file “hello.xml” to the master and add some different content than previous.**

**=**

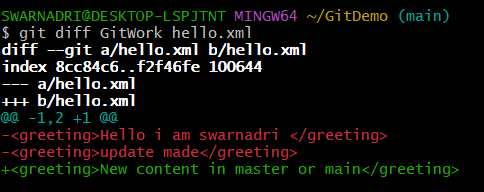
1. **Commit the changes to the master**

**=**

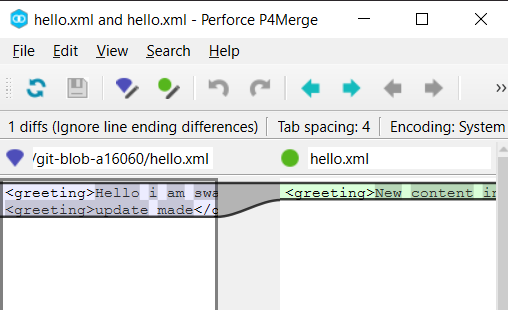
1. **Observe the log by executing “git log –oneline –graph –decorate –all”**

**=**

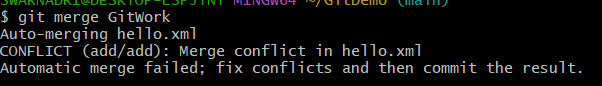
1. **Check the differences with Git diff tool**

**=**

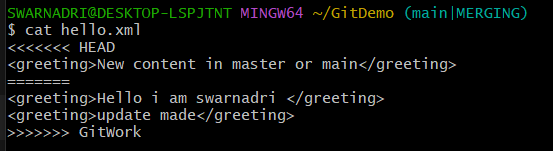
1. **For better visualization, use P4Merge tool to list out all the differences between master and branch**

**=**

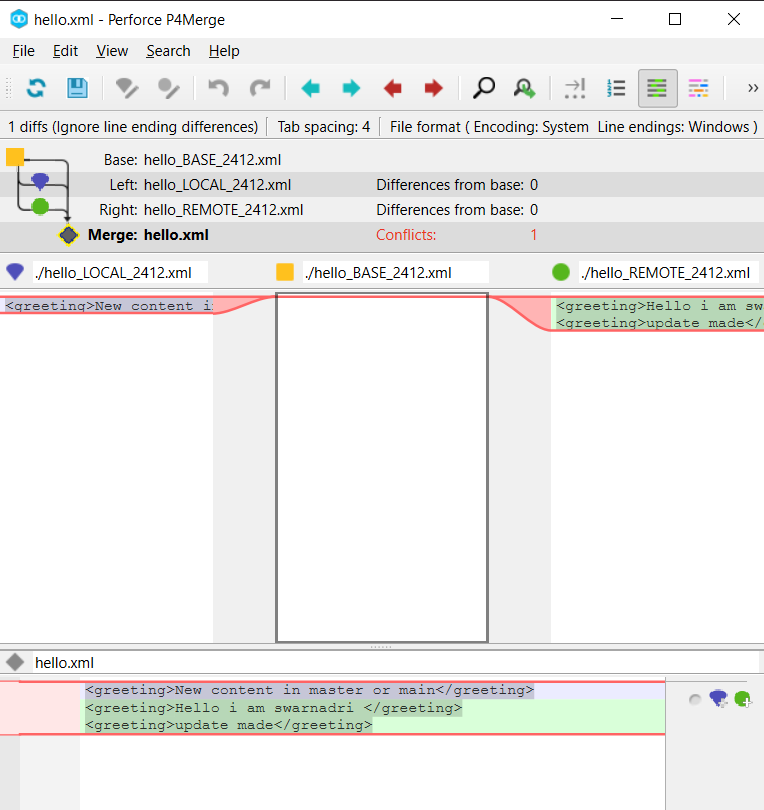
1. **Merge the bran to the master main**

**=**

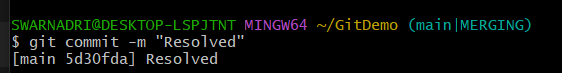
1. **Observe the git mark up.**

**=**

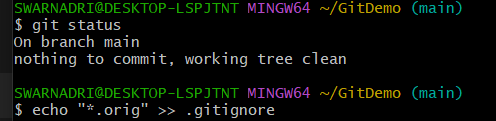
1. **Use 3-way merge tool to resolve the conflict**

**=**

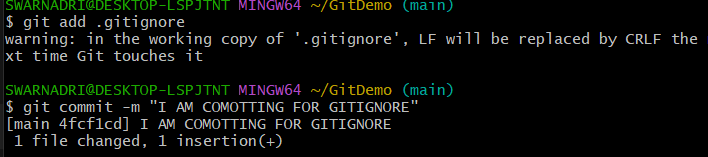
1. **Commit the changes to the master, once done with conflict**

**=**

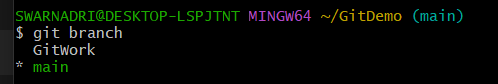
1. **Observe the git status and add backup file to the .gitignore file.**

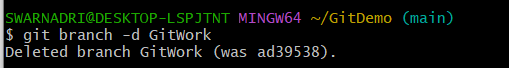
**=**

1. **Commit the changes to the .gitignore**

**=**

1. **List out all the available branches**

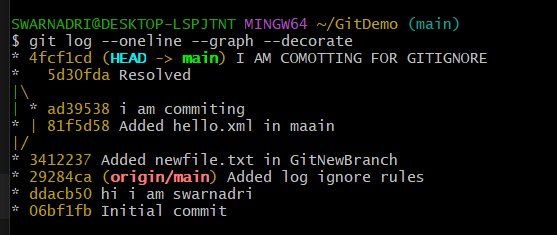
**=**

1. **Delete the branch, which merge to master.**

**=**

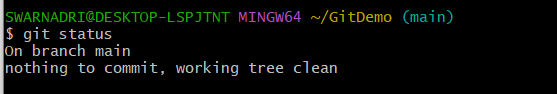
1. **Observe the log by executing “git log –oneline –graph –decorate”**

**=**

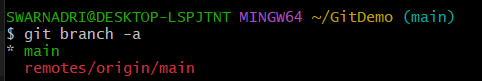
****

**HANDS ON – 5 ( file name: 5.Git – HOL)**

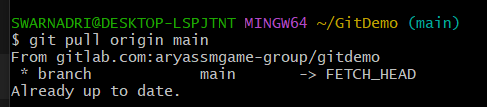
**Please follow the instructions to complete the hands-on. Each instruction expects a command  
for the Git Bash.  
1. Verify if master is in clean state.**

**=**

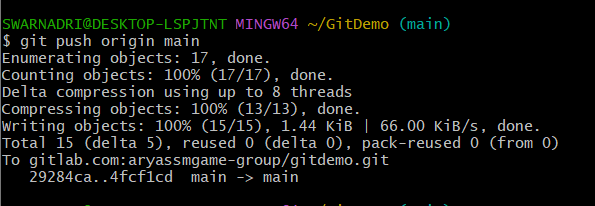
**2. List out all the available branches.**

**=**

**3. Pull the remote git repository to the master**

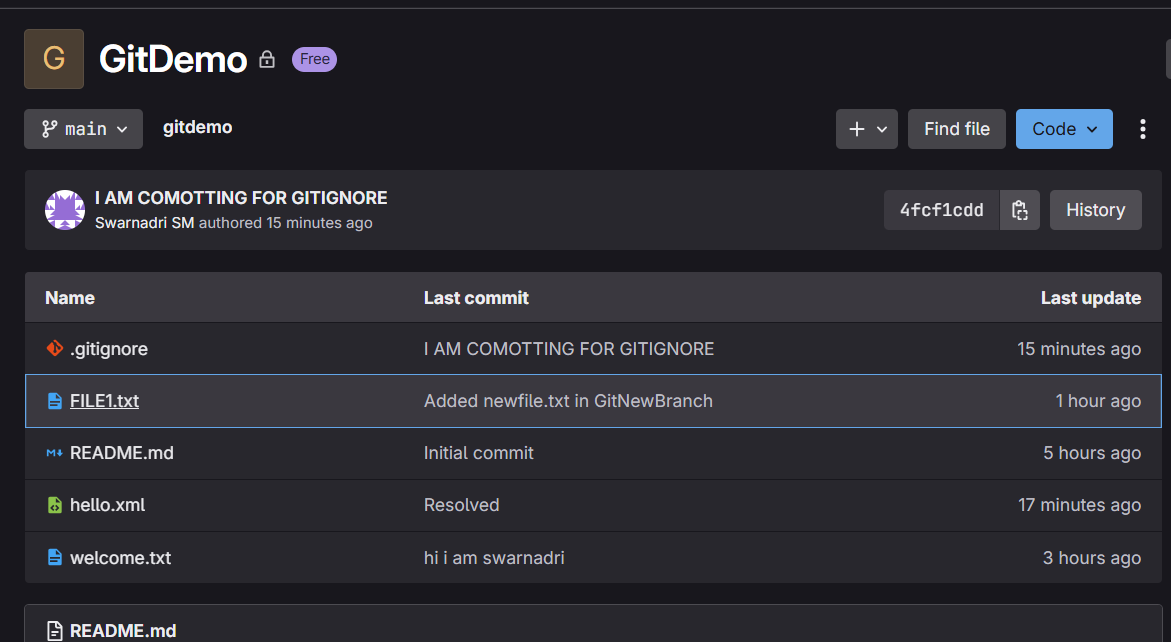
**=**

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

**=**

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

**=**

****

YES CHANGES ARE OBSERVED

**----------------------------------------------------------------------------------------------------**

**NAME: SWARNADRI SEKHAR MUKHERJEE**