**WEEK-8**

**GIT**

**1.GIT-HOL**

**Objectives**

**Familiar with Git commands like git init, git status, git add, git commit, git push, and git pull.**

**In this hands-on lab, you will learn how to**

* **Setup your machine with Git Configuration**
* **Integrate notepad++.exe to Git and make it a default editor**
* **Add a file to source code repository**

**CODE:**

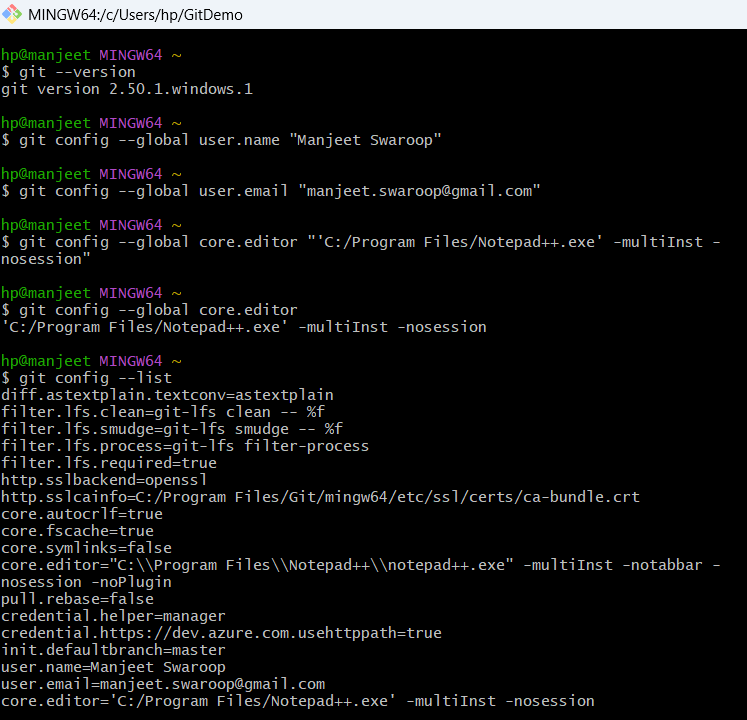
**Step 1: Setup your machine with Git Configuration**

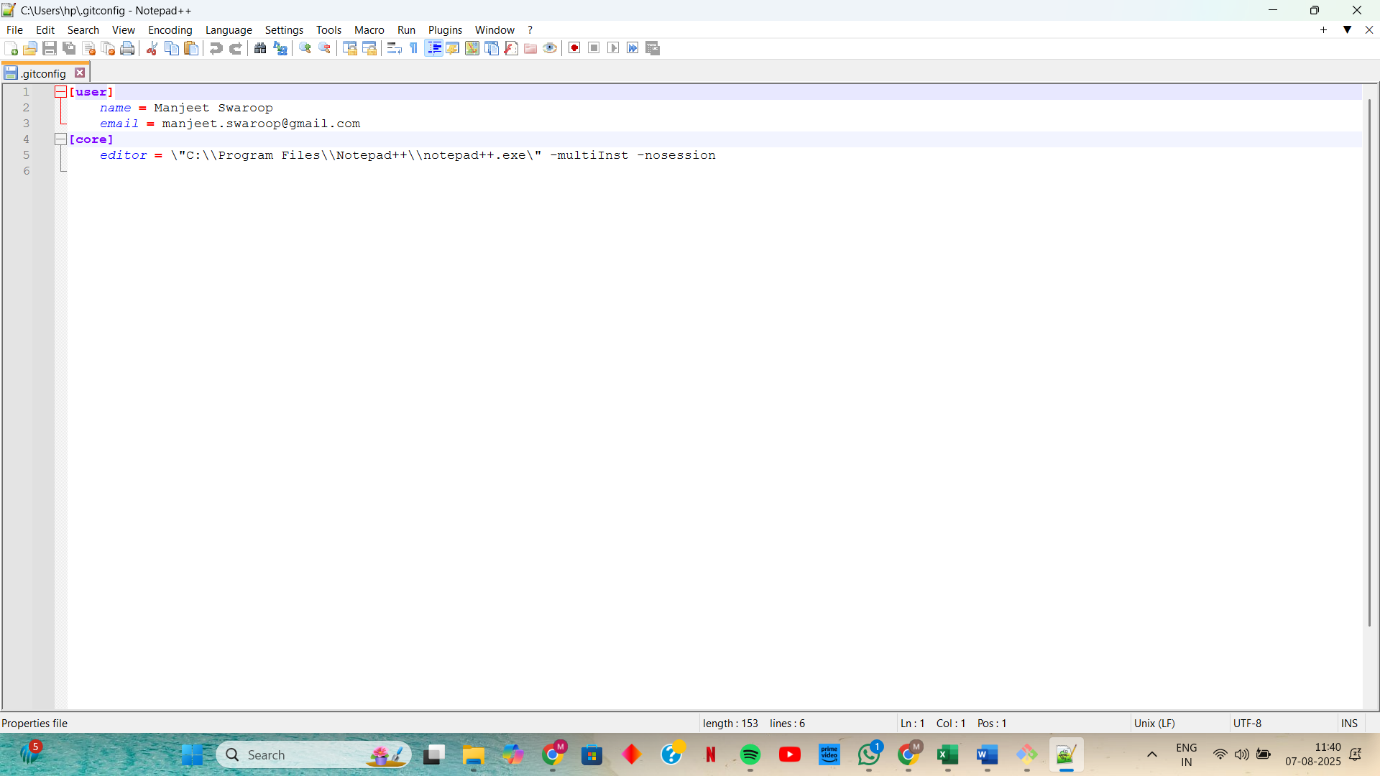
Check Git installation

Set Git global config

Verify config settings

**Step 2: Integrate Notepad++ with Git**





**Step 3: Add a File to Git Source Code Repository**

**Create a new local Git project** - git init GitDemo

cd GitDemo

**View hidden Git directory** - ls -al

**Create a new file** - echo "Welcome to the version control" >> welcome.txt

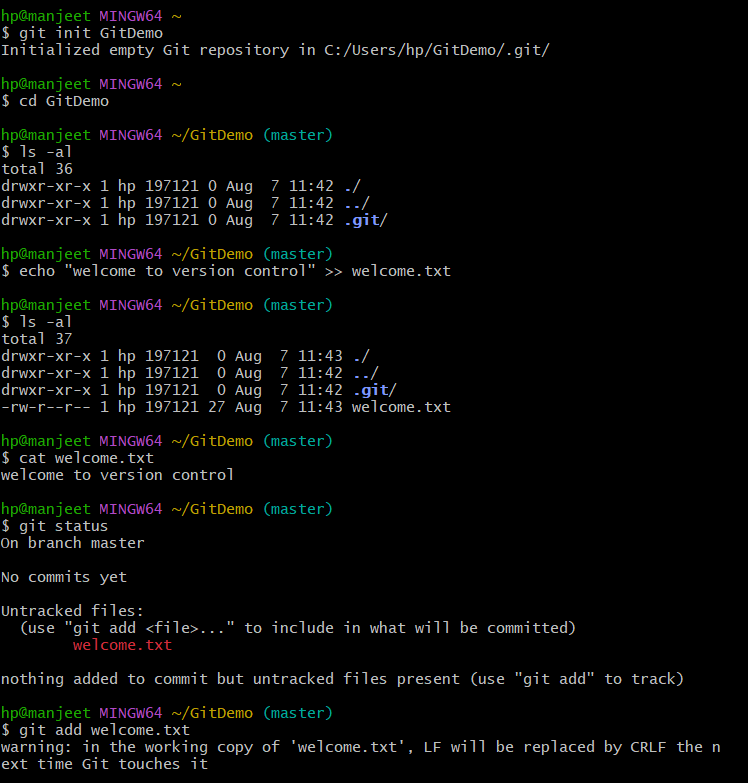
**Check file content** - cat welcome.txt

**Check Git status** - git status

**Track the file** - git add welcome.txt

**Commit the file** - git commit

**Check clean status** - git status

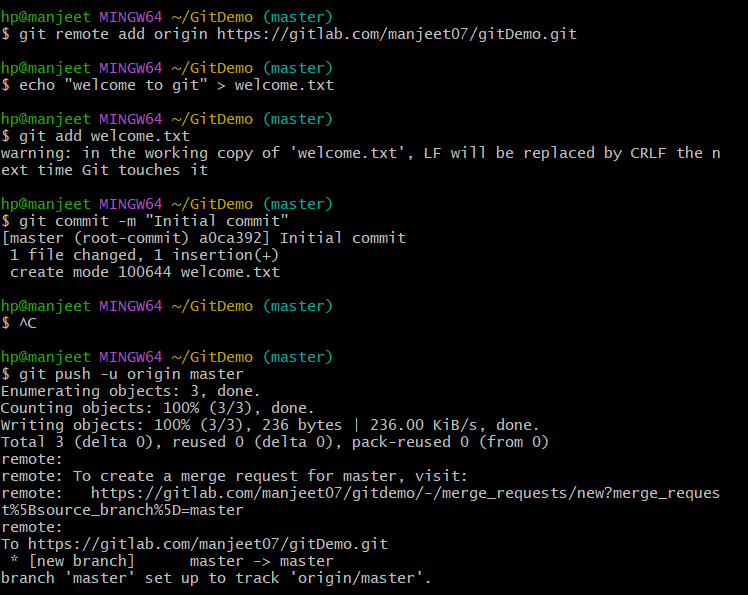
****

**Step 4: Push to Remote Repository (GitLab)**

Add remote origin

Push local repo to remote- git push origin master

**OUTPUT:**

****

**2.GIT-HOL**

**Objectives**

* **Explain git ignore**
* **Explain how to ignore unwanted files using git ignore**

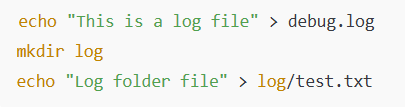
**In this hands-on lab, you will learn how to:**

* **Implement git ignore command to ignore unwanted files and folders**
* **Create a “.log” file and a log folder in the working directory of Git. Update .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.**

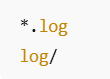
**CODE:**

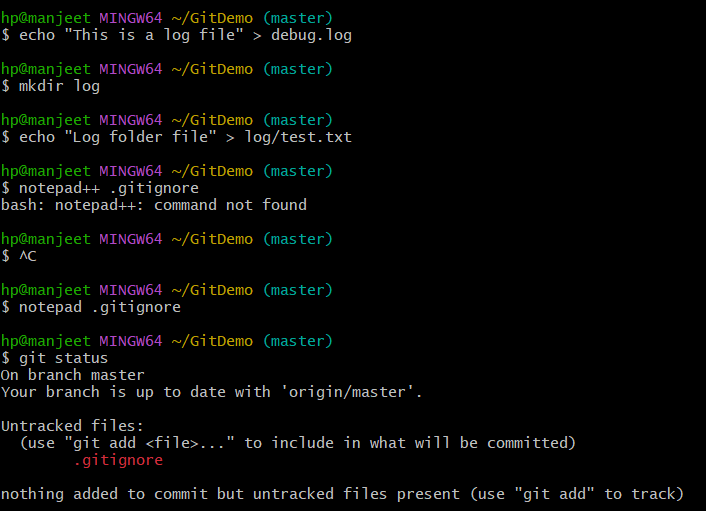
**Step 1: Go to Git working directory (like GitDemo):**

**Step 2: Create a .log file and a log/ folder**

****

**Step 3: Create and edit .gitignore file and inside add this**

****

****

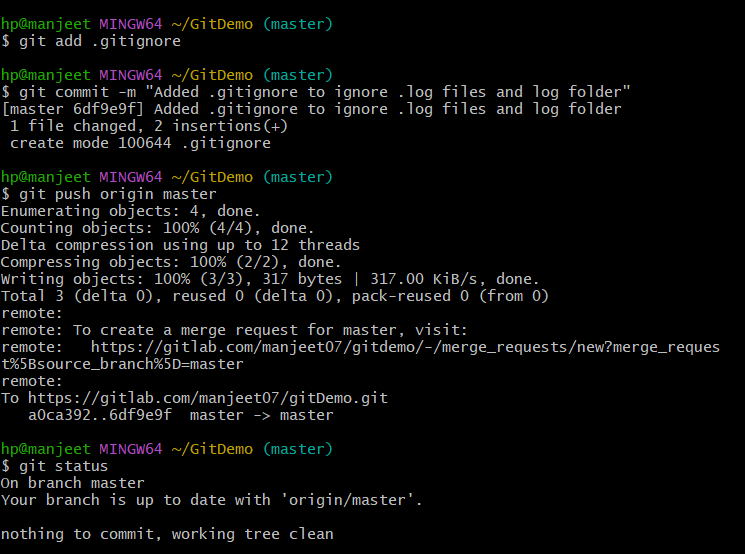
**Step 4: Check Git status**

**Step 5: Add and commit .gitignore**

**Step 6: Push the commit to GitLab**

**Step 7: Final Verification**

**OUTPUT:**

****

**3.GIT-HOL**

**Objectives**

* **Explain branching and merging**
* **Explain about creating a branch request in GitLab**
* **Explain about creating a merge request in GitLab**

**In this hands-on lab, you will learn how to:**

* **Construct a branch, do some changes in the branch, and merge it with master (or trunk)**

**Prerequisites**

* **Setting up Git environment with P4Merge tool for Windows**

**CODE:**

**Branching**

**Step 1: Create a new branch GitNewBranch -** git branch GitNewBranch

**Step 2: List all local and remote branches -** git branch -a

**Step 3: Switch to the new branch -** git checkout GitNewBranch

**Step 4: Add a file and make changes like this**

echo "This is a file in GitNewBranch" > branchfile.txt

git add branchfile.txt

git commit -m "Added branchfile.txt in GitNewBranch"

**Step 5: Check git status -** git status

**Merging**

**Step 6: Switch back to the master branch -** git checkout master

**Step 7: List differences between master and GitNewBranch-** git diff master GitNewBranch

**Step 8: View visual differences using P4Merge-** git difftool master GitNewBranch

**(Only works if you've set P4Merge as your diff tool)**

**If not set yet -** git config --global diff.tool p4merge

git config --global difftool.prompt false

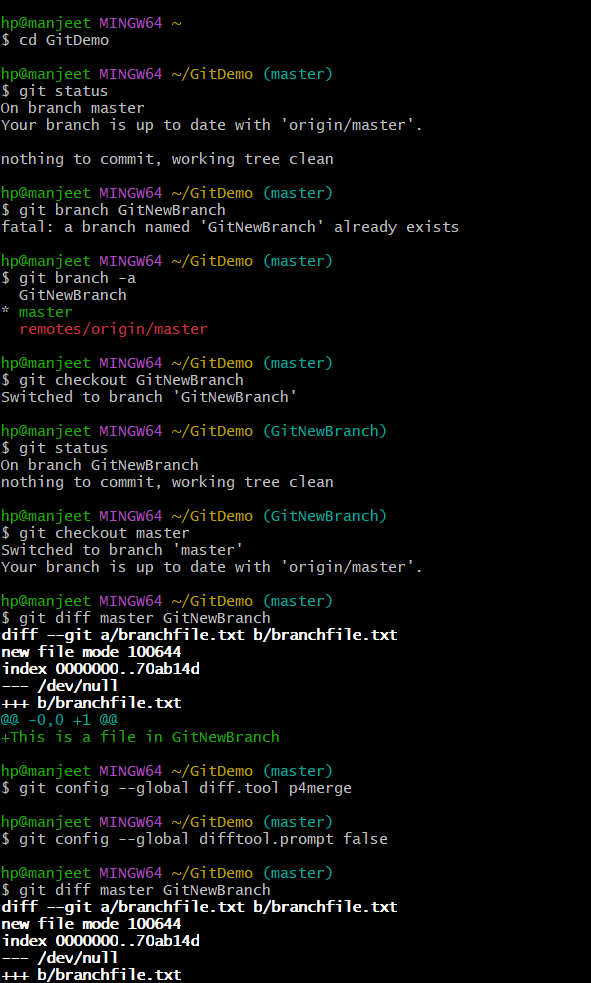
**Step 9: Merge the GitNewBranch into master -** git merge GitNewBranch

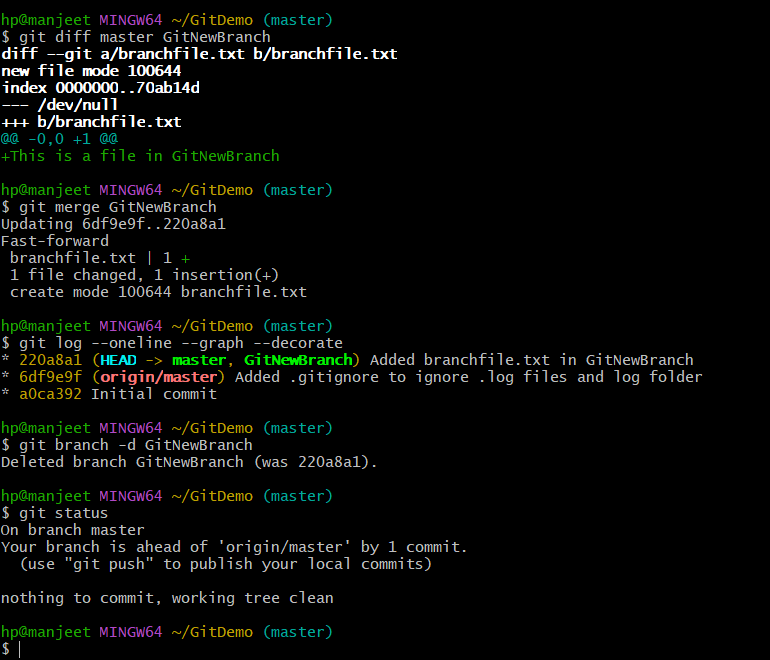
**Step 10: View commit history after merge -** git log --oneline --graph --decorate

**Step 11: Delete the branch-** git branch -d GitNewBranch

**Step 12: Final git status -** git status

**OUTPUT:**

****

****

**4.GIT-HOL**

**Objectives**

* **Explain how to resolve the conflict during merge.**

**In this hands-on lab, you will learn how to:**

* **Implement conflict resolution when multiple users are updating the trunk (or master) in such a way that it results into a conflict with the branch’s modification.**

**Prerequisites**

* **Hands-on ID: “Git-T03-HOL\_001”**

**CODE:**

**Step 1: Verify if master is in clean state**

git checkout master

git status

**Step 2: Create branch GitWork and add file hello.xml**

git checkout -b GitWork

echo "<msg>Hello from branch</msg>" > hello.xml

**Step 3: Update content of hello.xml and observe status**

echo "<branch>Change from GitWork branch</branch>" >> hello.xml

git status

**Step 4: Commit changes in branch**

git add hello.xml

git commit -m "GitWork: added and modified hello.xml"

**Step 5: Switch to master-** git checkout master

**Step 6: Add file hello.xml with different content**

echo "<msg>Hello from master</msg>" > hello.xml

echo "<master>Change from master branch</master>" >> hello.xml

**Step 7: Commit the changes to master**

git add hello.xml

git commit -m "Master: added and modified hello.xml"

**Step 8: Observe commit log**

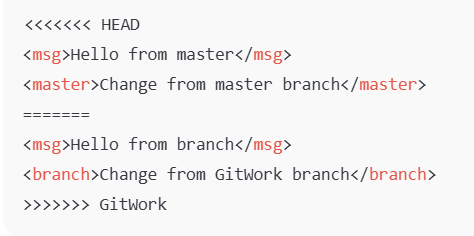
git log --oneline --graph --decorate --all

**Step 9: Check differences using CLI diff**

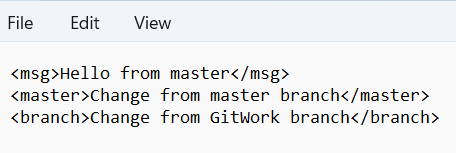
git diff GitWork

**Step 10: Visual diff using P4Merge -** git difftool GitWork

**Step 11: Merge branch into master (this will cause a conflict) -** git merge GitWork

****

**Step 12: Resolve conflict by manual**

****

**Step 13: Commit after resolving conflict**

git add hello.xml

git commit -m "Resolved conflict in hello.xml during merge of GitWork"

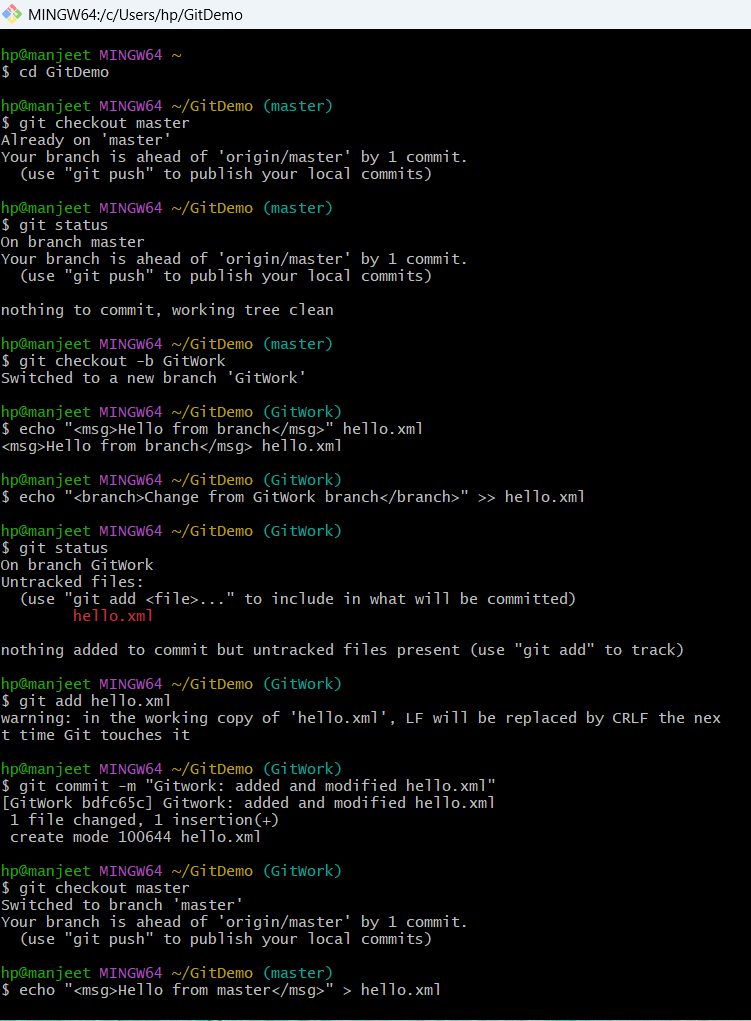
**Step 14: Check status and ignore backup files-** git status

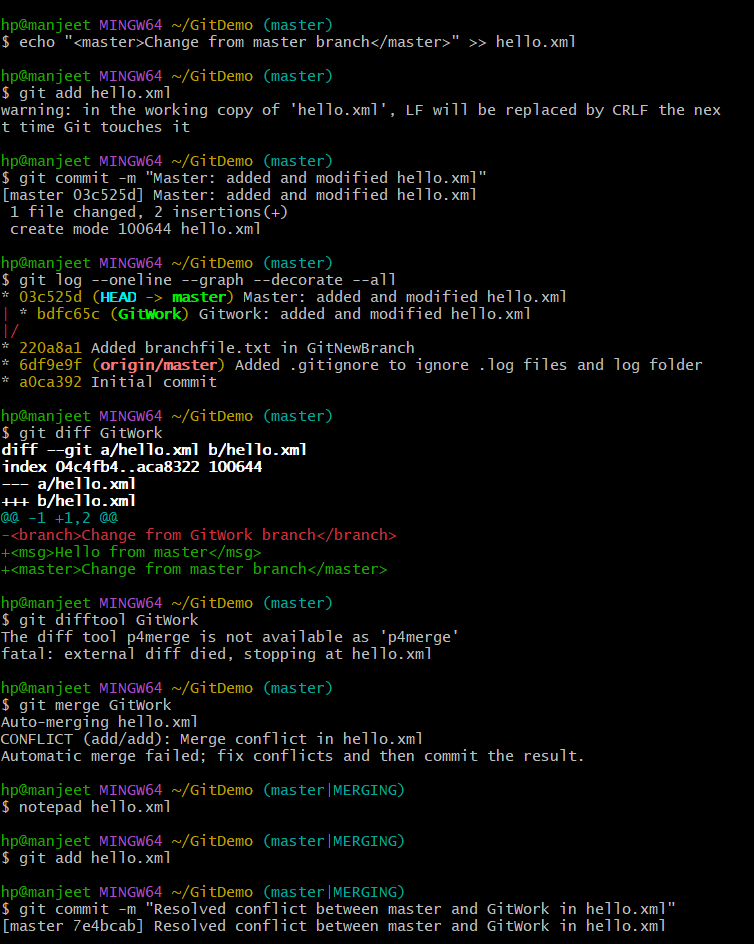
**Step 15: List all branches-** git branch

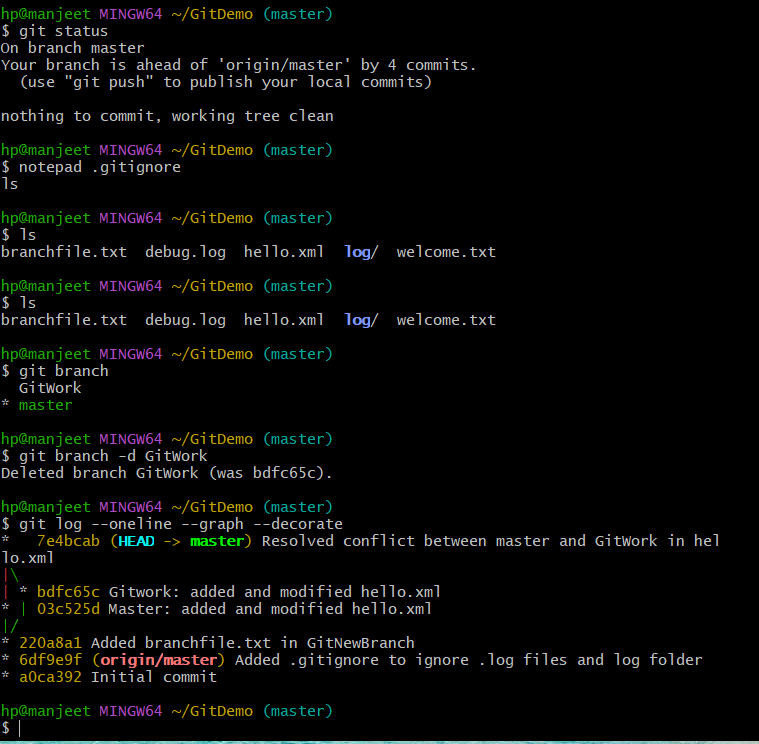
**Step 16: Delete the merged branch -** git branch -d GitWork

**Step 17: Final log view-** git log --oneline --graph --decorate

**OUTPUT:**

****

****

****

**5.GIT-HOL**

**Objectives**

* **Explain how to clean up and push back to remote Git**

**In this hands-on lab, you will learn how to:**

* **Execute steps involving clean up and push back to remote Git.**

**Prerequisites**

* **Hands-on ID: “Git-T03-HOL\_002”**

**CODE:**

**Step 1: Verify if master is in clean state –**

cd ~/GitDemo

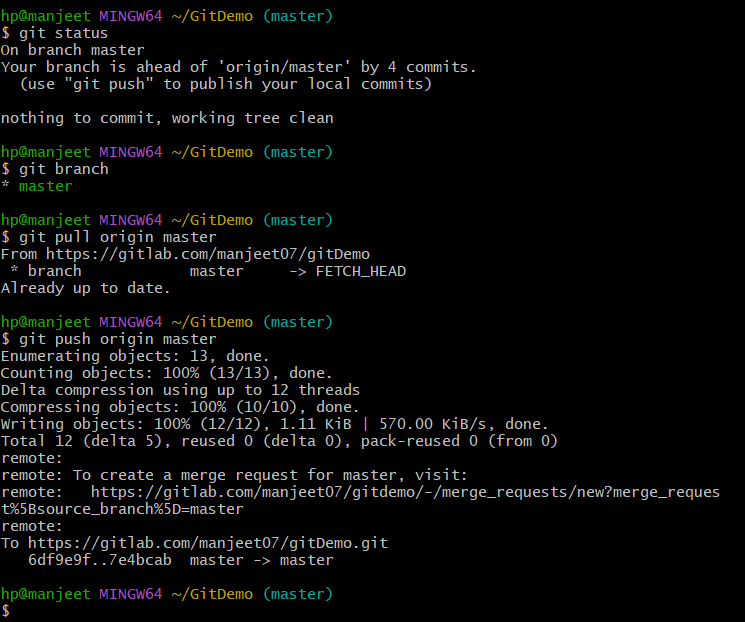
git status

**Step 2: List out all the available branches-** git branch

**Step 3: Pull the remote Git repository to the master-** git pull origin master

**Step 4: Push the pending changes to the remote repository-** git push origin master

**Step 5: Observe if the changes are reflected in the remote repository**

****

**OUTPUT:**

