**Digital Nurture 4.0 Week 8**

**GIT - HOL**

**Mandatory Hands On**

**File 1:GIT-HOL**

Step 1 – Git Configuration

1. Verify Git installation

git --version

git version 2.xx.x.windows.x

1. Set your Git username and email

git config --global user.name "Saranya"

git config --global user.email "ganeshsaranya2018@gmail.com"

1. Verify configuration

git config --list

user.name=Saranya

user.email=ganeshsaranya2018@gmail.com

Step 2 – Integrate Notepad++ as Default Editor

1. Check if Notepad++ runs

notepad++

1. Optional alias (shortcut)

alias np='notepad++'

1. Set Notepad++ as Git’s default editor

git config --global core.editor "'C:/Program Files/Notepad++/notepad++.exe' -multiInst -nosession"

1. Verify editor setting

git config --global -e

Step 3 – Create Project & Commit File

1. Create new directory and initialize Git

mkdir GitDemo

cd GitDemo

git init

Output:

Initialized empty Git repository in C:/path/to/GitDemo/.git/

1. Check .git folder exists

ls -a

1. Create a file

echo "Welcome to Git Demo" > welcome.txt

1. Check file

ls

cat welcome.txt

1. Check status

git status

1. Stage the file

git add welcome.txt

1. Commit the file (opens in Notepad++)

git commit

git commit -m "Initial commit: Added welcome.txt"

1. Verify

git status

Step 4 – Push to Remote GitLab

1. Create an empty repository on GitLab (same name: GitDemo).
2. Link local repo to remote

git remote add origin [Saranya\_81-group / GitDemo · GitLab](https://gitlab.com/saranya_81-group/gitdemo)

Pull remote changes

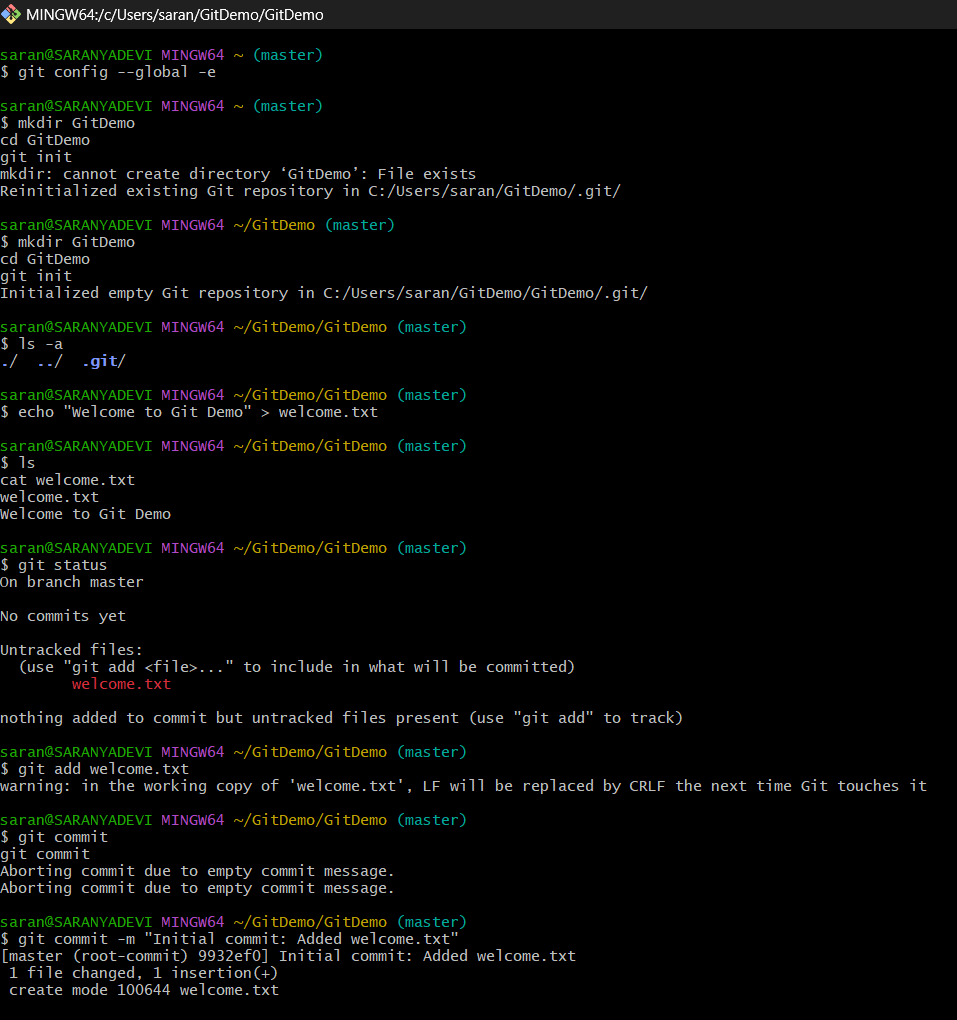
git pull origin master --allow-unrelated-histories

1. Push local commits to remote

git push -u origin master

**OUTPUT:**

****

****

**File 2:GIT-HOL**

Objectives

* Learn what .gitignore is.
* Learn how to ignore unwanted files or folders in Git.

**How to Ignore Files Using**

**.gitignore**

You **list patterns** in .gitignore for the files you want Git to ignore.

\*.log

log/

STEPS:

**Step 1 – Create a .log file and a log folder**

echo "This is a log file" > error.log

mkdir log

echo "Log details here" > log/debug.txt

Now your folder looks like:

lua

GitDemo/

error.log

log/debug.txt

**Step 2 – Create/Edit .gitignore**

notepad++ .gitignore

Add these lines:

\*.log

log/

Save and close.

**Step 3 – Check Git Status**

git status

**Step 4 – Commit Changes**

1. Add .gitignore:

git add .gitignore

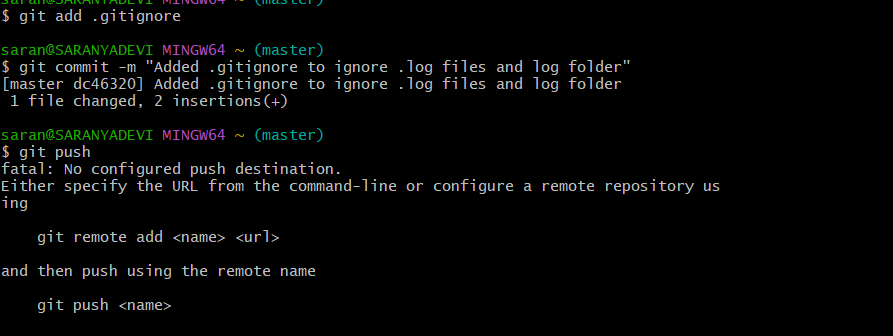
1. Commit:

git commit -m "Added .gitignore to ignore .log files and log folder"

1. Push:

git push

**OUTPUT:**



**File 3:GIT-HOL**

**Step 1 – Branching**

1. **Create a new branch GitNewBranch**

git branch GitNewBranch

**2. List all branches (local & remote)**

git branch -a

* The branch with \* is your **current** branch.

**3. Switch to the new branch**

git checkout GitNewBranch

git switch GitNewBranch

**4. Add some files to the branch**

echo "This is content for GitNewBranch" > branchfile.txt

**5. Stage and commit changes**

git add branchfile.txt

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

**6. Check status**

git status

**Step 2 – Merging**

**1. Switch to master branch**

git checkout master

**2. List differences between master and branch**

git diff master GitNewBranch

This shows changes in the command-line format.

**3. See visual differences using P4Merge**

git difftool master GitNewBranch

**4. Merge GitNewBranch into master**

git merge GitNewBranch

**5. View merge history**

git log --oneline --graph --decorate

**6. Delete the branch after merging**

git branch -d GitNewBranch

* -d deletes only if merged,
* -D forces deletion even if unmerged.

**7. Check status**

git status

**Step 3 – Creating Merge Request in GitLab**

1. **Push your branch to GitLab**

git push -u origin GitNewBranch

1. **Open GitLab in browser**

* Go to your project → You’ll see a message like:  
  *“GitNewBranch has been pushed. Create merge request?”*
* Click **"Create merge request"**

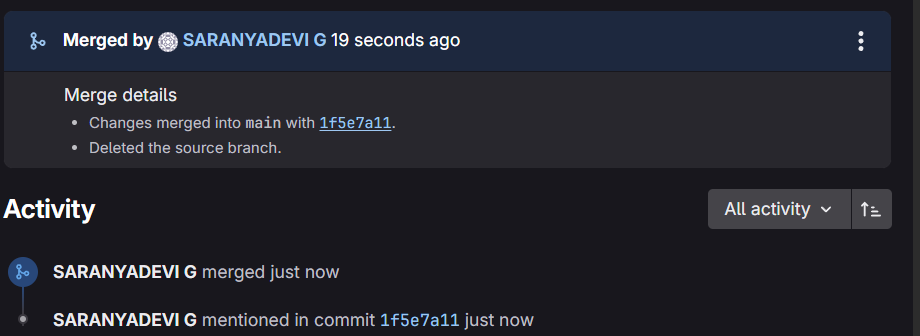
1. **Fill Merge Request details**

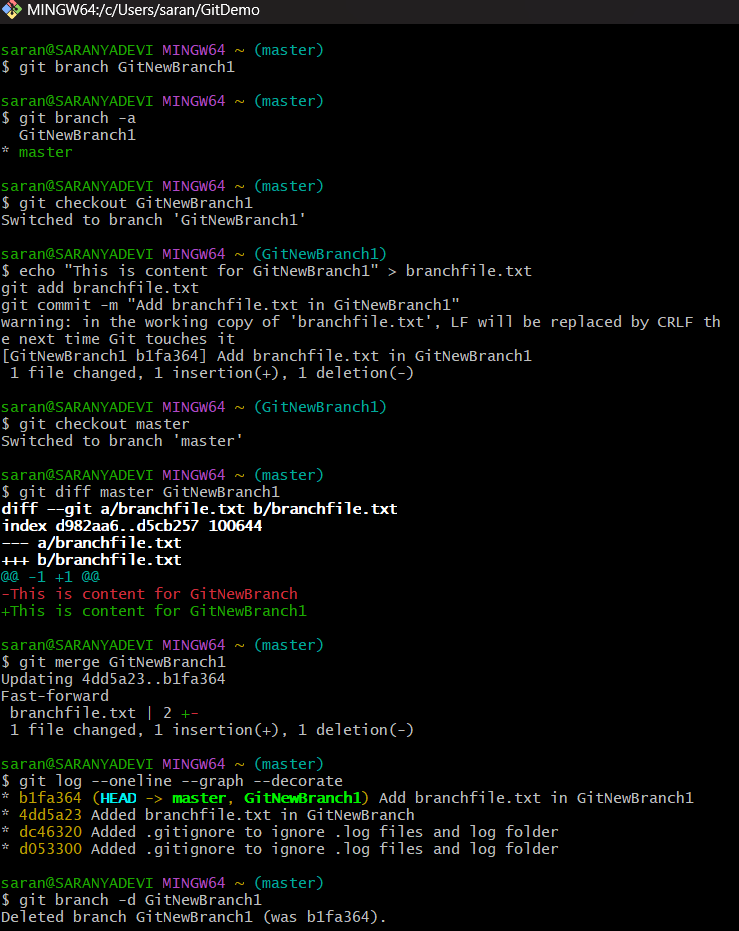
* Title: *Merge GitNewBranch into master*
* Description: Explain your changes.
* Reviewer: Select someone (or leave blank if not needed).
* Click **Submit Merge Request**.

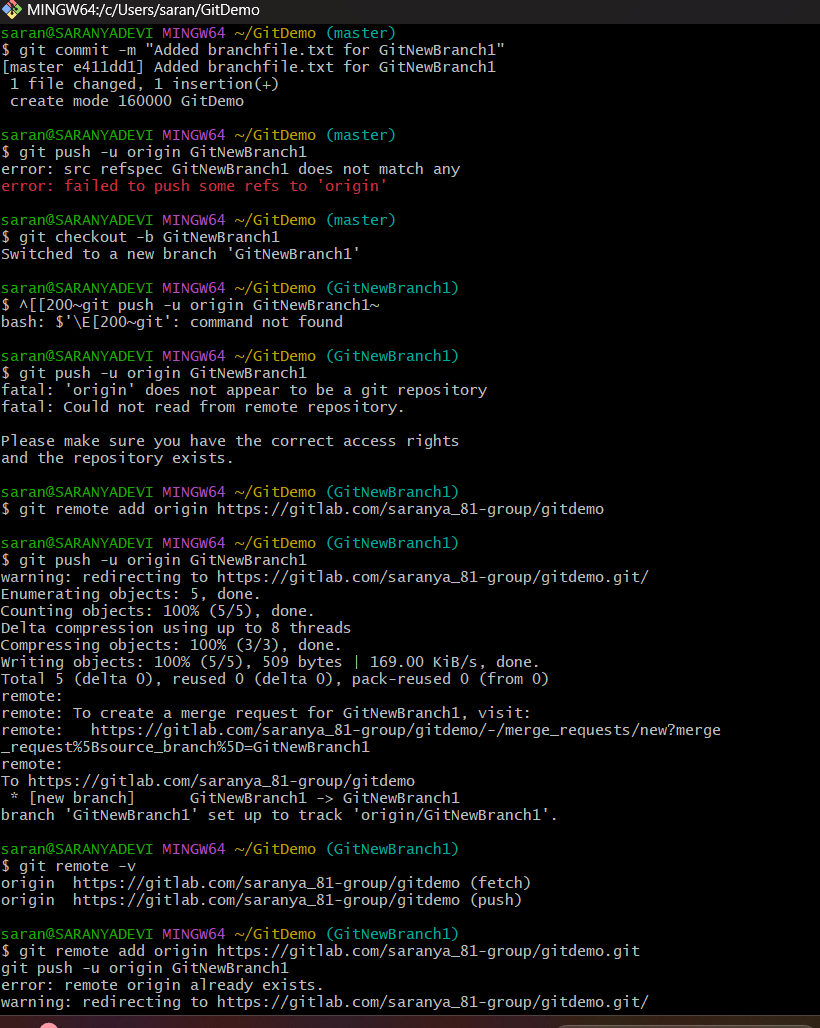
1. **Approve & Merge**

* If approvals are not required, you can merge directly.

**OUTPUT:**







**File 4:GIT-HOL**

**1. Verify if master is in a clean state**git checkout master

git status

You should see:

On branch master

nothing to commit, working tree clean

**2. Create a branch GitWork and add a file hello.xml**

git checkout -b GitWork

echo "<message>Hello from GitWork branch</message>" > hello.xml

git status

**3. Update the content of hello.xml**

echo "<message>Updated content in GitWork branch</message>" > hello.xml

git status

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

git add hello.xml

git commit -m "Add hello.xml in GitWork branch"

**5. Switch to master**

git checkout master

**6. Add hello.xml to master with different content**

echo "<message>Hello from master branch</message>" > hello.xml

**7. Commit the changes to master**

git add hello.xml

git commit -m "Add hello.xml in master branch"

**8. Observe the log**

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

**9. Check differences using Git diff**

git diff GitWork

**10. (Optional) Visualize with P4Merge**

git mergetool --tool=p4merge

**11. Merge the branch into master**

git merge GitWork

**12. Observe Git conflict markers**

Open hello.xml

<<<<<<< HEAD

<message>Hello from master branch</message>

=======

<message>Updated content in GitWork branch</message>

>>>>>>> GitWork

**13. Resolve conflict using a 3-way merge tool**

If have a merge tool:

git mergetool

**14. Commit the resolved merge**

git add hello.xml

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

**15. Add backup files to .gitignore**

echo "\*.orig" >> .gitignore

git status

**16. Commit .gitignore**

git add .gitignore

git commit -m "Add .gitignore to exclude backup files"

**17. List all branches**

git branch

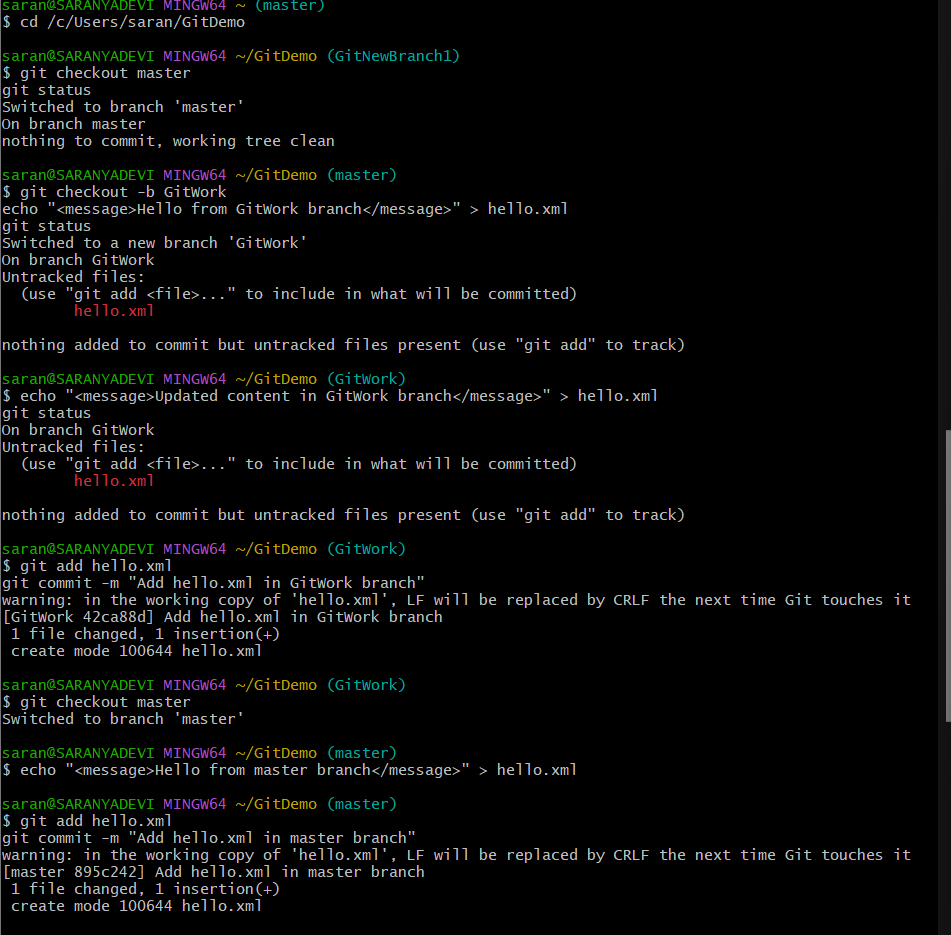
**18. Delete the merged branch**

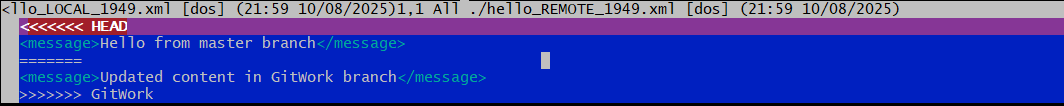
git branch -d GitWork

**19. View final log**

git log --oneline --graph --decorate

**OUTPUT:**

****

****

**File 5:GIT-HOL**

**1. Verify if master is in a clean state**

git status

* If it says **"working tree clean"**, you’re good.
* If not, you’ll see:
  + **Modified files** → git add <file> then git commit -m "Your message"
  + **Untracked files** → git add <file> then git commit -m "Your message"

**2. List all available branches**

git branch -a

* \* marks the branch you’re currently on.
* If you’re not on master, switch:

git checkout master

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

git pull origin master

This ensures your local master is up to date.

**4. Push the pending changes from Git-T03-HOL\_002**

git checkout master

git merge Git-T03-HOL\_002

git add .

git commit -m "Merged Git-T03-HOL\_002 into master"

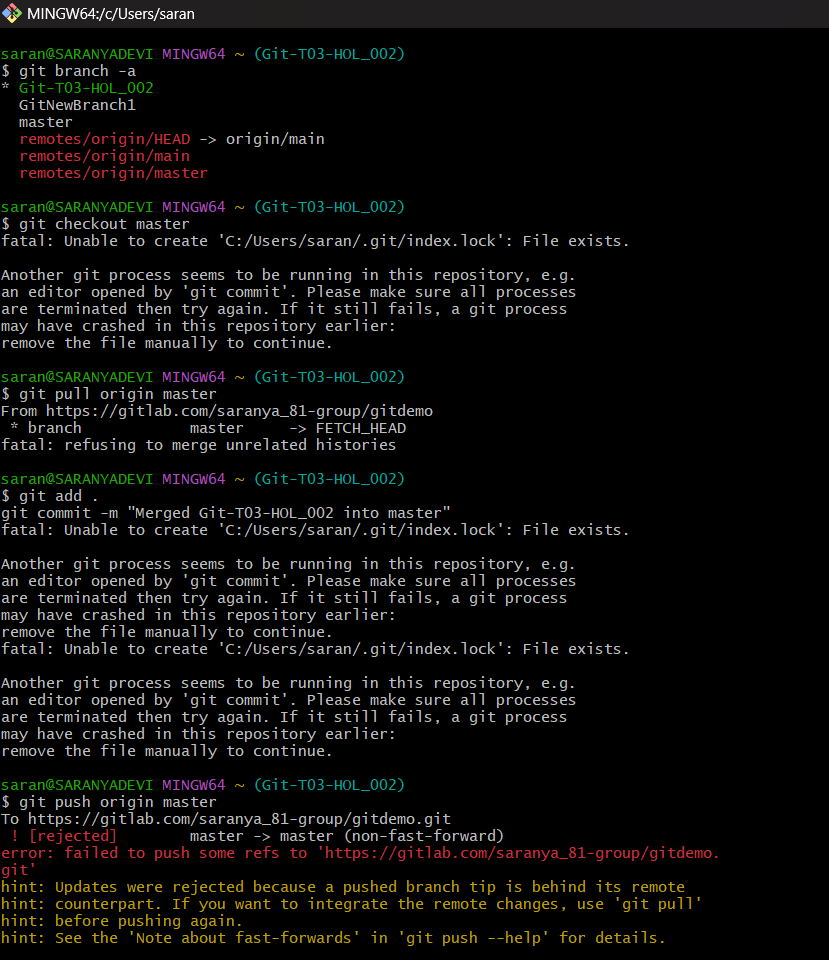
Finally, push:

git push origin master

**5. Verify on remote**

* Go to your **GitHub/GitLab repository** in the browser.
* Refresh the page and confirm your changes appear.

**OUTPUT:**



****