Week8\_Handson

1. Git-HOL

**Step 1: Git Configuration**

1. Open Git Bash and check if Git is installed:  
 > git --version  
  
2. Configure your Git user name and email:  
 > git config --global user.name "Sanjana Shetty"  
 > git config --global user.email "sanjanabalu04@gmail.com"  
  
3. Verify the configuration:  
 > git config --global --list  
 Expected output:  
 user.name=Sanjana Shetty  
 user.email=sanjanabalu04@gmail.com

**Step 2: Set Notepad++ as Default Git Editor**

1. Install Notepad++ from: https://notepad-plus-plus.org/downloads/  
  
2. Add Notepad++ to PATH:  
 - Go to: Control Panel > System > Advanced system settings > Environment Variables  
 - Under "User Variables", edit the 'Path' variable and add:  
 C:\Program Files\Notepad++\  
  
3. Close and reopen Git Bash. Test if it works:  
 > notepad++  
  
4. Set Notepad++ as Git's default editor:  
 > git config --global core.editor "notepad++ -multiInst -notabbar -nosession -noPlugin"  
  
5. Verify:  
 > git config --global --list  
 Expected output includes:  
 core.editor=notepad++ -multiInst -notabbar -nosession -noPlugin

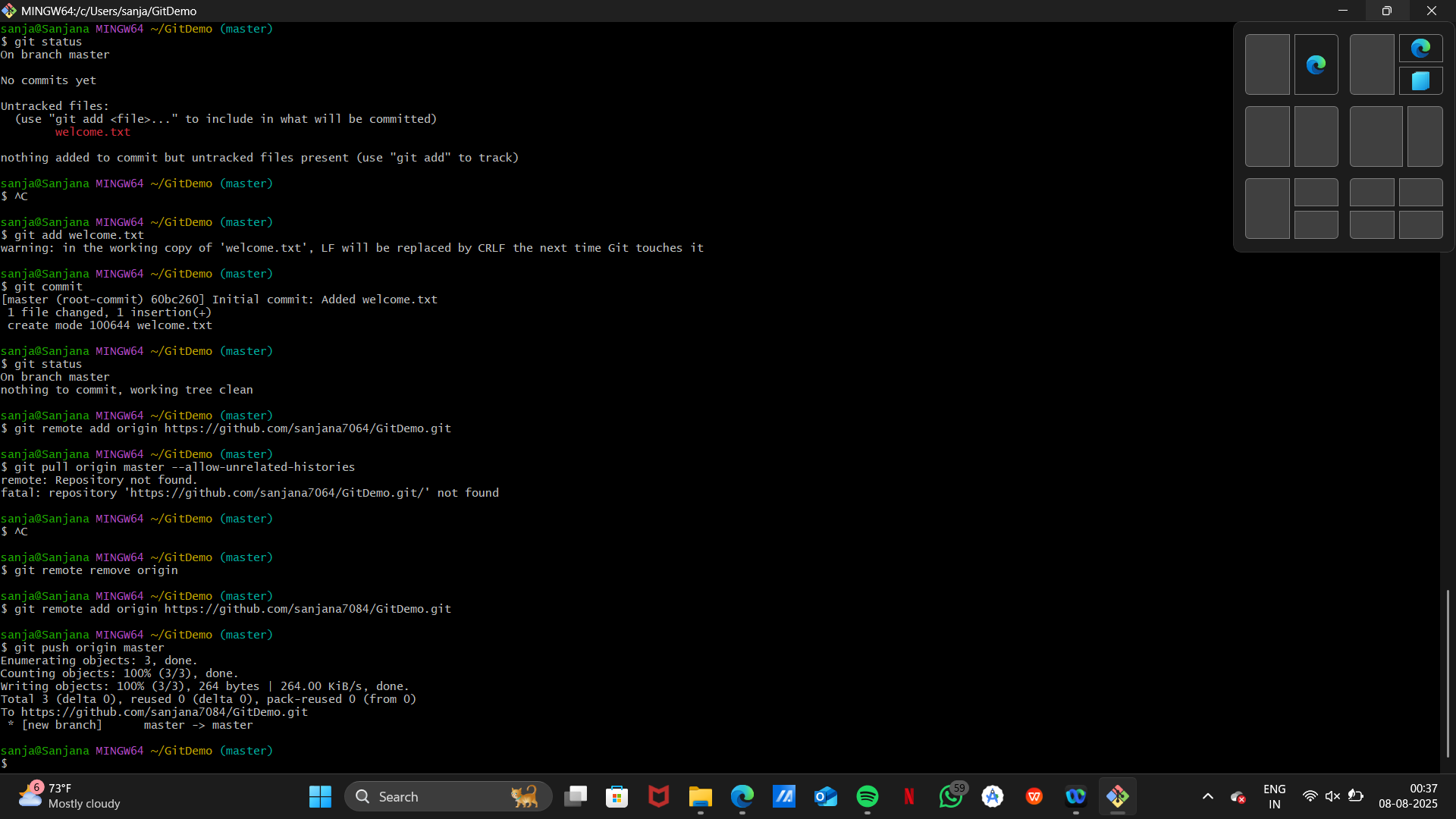
**Step 3: Initialize Git Repository and Commit**

1. Open Git Bash in your home folder or a chosen directory.  
  
2. Create a folder and initialize Git:  
 > mkdir GitDemo  
 > cd GitDemo  
 > git init  
  
3. Create and commit a file:  
 > echo "Welcome to Git hands-on lab" > welcome.txt  
 > git add welcome.txt  
 > git commit  
 (Write commit message in Notepad++, save and close)  
  
4. Verify status:  
 > git status  
 Expected: nothing to commit, working tree clean

**Step 4: Push to GitHub**

1. Go to GitHub.com and create a new repository named 'GitDemo'. DO NOT initialize with a README.  
  
2. Back in Git Bash, connect the local repo to GitHub:  
 > git remote add origin https://github.com/sanjana7064/GitDemo.git  
  
3. If needed, remove incorrect origin first:  
 > git remote remove origin  
  
4. Pull any remote data (if needed):  
 > git pull origin master --allow-unrelated-histories  
  
5. Push to GitHub:  
 > git push origin master

Output:



2. Git-HOL

**Step 1: Open Your Git Project Folder**

Open Git Bash and navigate to your project folder:  
  
> cd ~/GitDemo

**Step 2: Create Files and Folder to be Ignored**

Create a .log file and a folder named log with a file inside:  
  
> echo "This is a log file" > debug.log  
> mkdir log  
> echo "Log folder content" > log/log.txt  
  
Then check what was created:  
  
> ls  
> ls log

**Step 3: Create .gitignore File**

Open the .gitignore file using Notepad++:  
  
> notepad++ .gitignore  
  
Add the following lines and save:  
  
\*.log  
log/

**Step 4: Check Git Status**

Check Git status to ensure .log files and log/ folder are ignored:  
  
> git status  
  
Expected: Only .gitignore is visible as a new file.

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

Stage and commit the .gitignore file:  
  
> git add .gitignore  
> git commit  
  
Enter commit message in Notepad++:  
Added .gitignore to ignore log files and folders

**Step 6: Verify .gitignore is Working**

Create additional test files and check status:  
  
> echo "Another log file" > error.log  
> mkdir log2  
> echo "Test" > log2/logfile.txt  
  
> git status

**3. Git-HOL**

**1. Create a new branch**

git branch GitNewBranch

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

git branch -a

\* indicates the current branch.

**3. Switch to the new branch:**

git checkout GitNewBranch

**4. Create a file and add content:**

echo "Branch content here" > branchfile.txt

**5. Stage and commit the file:**

git add branchfile.txt

git commit -m "Added file in GitNewBranch"

**6. Check Git status:**

git status

**Merging Steps**

**1. Switch back to master:**

git checkout master

**2. Show command-line differences:**

git diff GitNewBranch

**3. Show visual diff (with P4Merge configured):**

git difftool GitNewBranch

**4. Merge the branch into master:**

git merge GitNewBranch

**5. View merge history:**

git log --oneline --graph --decorate

**6. Delete the branch after merge:**

git branch -d GitNewBranch

**7. Check final status:**

git status.

**4. Git-HOL**  
1. Verify if master is clean:  
 > git status  
  
2. Create a new branch and add a file:  
 > git checkout -b GitWork  
 > echo "<message>Hello from branch</message>" > hello.xml  
  
3. Check status and commit the file:  
 > git status  
 > git add hello.xml  
 > git commit -m "Added hello.xml in GitWork"  
  
4. Switch to master:  
 > git checkout master  
  
5. Create a conflicting version of hello.xml:  
 > echo "<note>Hello from master</note>" > hello.xml  
  
6. Commit the master version:  
 > git add hello.xml  
 > git commit -m "Added conflicting hello.xml in master"  
  
7. View the logs visually:  
 > git log --oneline --graph --decorate --all  
  
8. Check diff from branch to master:  
 > git diff GitWork  
  
9. For visual diff, use P4Merge:  
 > git difftool GitWork  
  
10. Attempt to merge GitWork into master:  
 > git merge GitWork  
  
11. Git will detect a conflict and mark the conflict in hello.xml  
  
12. Open the file in a merge tool (e.g., P4Merge) and resolve conflict manually  
  
13. After resolving, mark the file as resolved:  
 > git add hello.xml  
  
14. Commit the resolved changes:  
 > git commit -m "Resolved merge conflict in hello.xml"  
  
15. Check status:  
 > git status  
  
16. Add backup files to .gitignore (if any like hello.xml.orig):  
 > notepad++ .gitignore  
 (Add line: \*.orig)  
  
17. Stage and commit the .gitignore file:  
 > git add .gitignore  
 > git commit -m "Ignored backup merge files"  
  
18. List all branches:  
 > git branch  
  
19. Delete the merged branch:  
 > git branch -d GitWork  
  
20. View the final log:  
 > git log --oneline --graph --decorate

**5. Git-HOL**

**1. Verify that the master branch is in a clean state:**

**git status**

**2. List all available branches:**

**git branch -a**

**3. Pull the latest updates from the remote:**

**git pull origin master**

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

**git push origin master**

**5. Confirm on GitHub or GitLab:**

* **Open your repository in the browser**
* **Check if the new commits or file changes are visible**