**HandsOn 1:**

# Step 1: Check Git installation

git --version

# Configure Git username & email

git config --global user.name "Your Name"

git config --global user.email "youremail@example.com"

# Verify configuration

git config –list

# Step 2: Check if Notepad++ runs

notepad++

# Add alias for Notepad++

alias np='notepad++'

# Make alias permanent

notepad++ ~/.bash\_profile

# Add: alias np='notepad++'

# Set Notepad++ as default editor

git config --global core.editor "notepad++ -multiInst -nosession"

# Verify editor

git config --global -e

# Step 3: Create and initialize repo

mkdir GitDemo

cd GitDemo

git init

# Verify .git folder

ls -a

# Create file

echo "Welcome to Git Demo" > welcome.txt

# Verify file

ls

# View file content

cat welcome.txt

# Check status

git status

# Stage file

git add welcome.txt

# Commit (opens editor)

git commit

# Check status again

git status

# Step 4: Link to GitLab & push

git remote add origin https://gitlab.com/USERNAME/GitDemo.git

git pull origin master

git push origin master

**HandsOn 2:**

# Create a log folder and a .log file

mkdir log

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

echo "This is a log inside folder" > log/test.log

# Create .gitignore file

echo "\*.log" > .gitignore

echo "log/" >> .gitignore

# Verify .gitignore contents

cat .gitignore

# Check status (log files/folder should appear under 'untracked' before committing .gitignore)

git status

# Stage and commit only .gitignore

git add .gitignore

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

# Check status again (log files/folder should now be ignored)

git status

**HandsOn 3:**

# 1. Create a new branch

git branch GitNewBranch

# 2. List all branches (local and remote)

git branch -a

# 3. Switch to the new branch

git checkout GitNewBranch

# Add some files with content

echo "File for GitNewBranch" > branchfile1.txt

echo "Another file for GitNewBranch" > branchfile2.txt

# 4. Commit the changes to the branch

git add branchfile1.txt branchfile2.txt

git commit -m "Added files in GitNewBranch"

# 5. Check status

git status

# 1. Switch to master

git checkout master

# 2. List differences between master and GitNewBranch

git diff master GitNewBranch

# 3. (Optional) View visual differences using P4Merge

git mergetool --tool=p4merge

# 4. Merge GitNewBranch into master

git merge GitNewBranch

# 5. View merge history

git log --oneline --graph --decorate

# 6. Delete the branch

git branch -d GitNewBranch

# Check status

git status

**HandsOn 4:**

# 1. Verify if master is in clean state

git checkout master

git status

# 2. Create a branch “GitWork” and add a file

git branch GitWork

git checkout GitWork

echo "<greeting>Hello from GitWork</greeting>" > hello.xml

# 3. Update the content of “hello.xml” and observe status

echo "<greeting>Hello again from GitWork</greeting>" >> hello.xml

git status

# 4. Commit the changes

git add hello.xml

git commit -m "Added and updated hello.xml in GitWork"

# 5. Switch to master

git checkout master

# 6. Add a file “hello.xml” with different content

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

# 7. Commit the changes to master

git add hello.xml

git commit -m "Added hello.xml in master with different content"

# 8. Observe the log

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

# 9. Check differences

git diff master GitWork

# 10. View differences in P4Merge

git mergetool --tool=p4merge

# 11. Merge GitWork into master

git merge GitWork

# 12. Observe merge status

git status

# 13. Resolve conflicts using 3-way merge tool

git mergetool

# 14. Commit after resolving conflict

git add hello.xml

git commit -m "Resolved merge conflict between master and GitWork"

# 15. Observe git status and add backup file to .gitignore

git status

echo "\*~" >> .gitignore

# 16. Commit the changes to .gitignore

git add .gitignore

git commit -m "Ignore backup files"

# 17. List all branches

git branch -a

# 18. Delete the merged branch

git branch -d GitWork

# 19. Observe the log

git log --oneline --graph –decorate

**HandsOn 5:**

# 1. Verify if master is in clean state

git checkout master

git status

# 2. List out all available branches

git branch -a

# 3. Pull the remote git repository to master

git pull origin master

# 4. Push changes from “Git-T03-HOL\_002” to remote

git checkout Git-T03-HOL\_002

git push origin Git-T03-HOL\_002

# 5. (Optional) Verify changes on remote (opens in browser)

# Replace URL with your repo link

git remote -v