1. Git-HOL

Step 1: Setup Your Machine with Git Configuration

git --version(to check if Git is installed)

Configure your Git user info (username and email)

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

git config --global user.email ["your.email@example.com"](mailto:\"your.email@example.com\")

git config --list

Step 2: Integrate Notepad++ with Git

notepad++

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

git config --global -e

Step 3: Add a File to a Git Repository

ls -a

echo "Welcome to Git Hands-on Lab!" > welcome.txt

cat welcome.txt

git status

git add welcome.txt

git commit

Added welcome.txt

- Initial commit

- Contains greeting text

Step 4: Connect to Remote Repository (GitLab)

git remote add origin <https://gitlab.com/yourusername/GitDemo.git>

git pull origin master --allow-unrelated-histories

git push origin master

1. Git-HOL

cd GitDemo

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

mkdir log

echo "Log content" > log/logfile.txt

git status

notepad++ .gitignore

nano .gitignore

\*.log

log/

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

git status

1. Git-HOL

git checkout -b GitNewBranch

git branch -a

echo "Branch-specific changes" > branchfile.txt

git add branchfile.txt

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

git status

git checkout master

git diff GitNewBranch

git config --global diff.tool p4merge

git config --global merge.tool p4merge

git config --global mergetool.p4merge.cmd '"C:/Program Files/Perforce/p4merge.exe" "$LOCAL" "$REMOTE"'

git difftool GitNewBranch

git merge GitNewBranch

git log --oneline --graph --decorate

git branch -d GitNewBranch

git status

git push origin --delete GitNewBranch

1. Git-HOL

git checkout master

git status

git checkout -b GitWork

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

echo "<note>Conflict demo content</note>" >> hello.xml

git add hello.xml

git commit -m "Added hello.xml in GitWork with sample content"

git checkout master

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

git add hello.xml

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

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

git diff GitWork

git config --global merge.tool p4merge

git config --global mergetool.p4merge.cmd '"C:/Program Files/Perforce/p4merge.exe" "$BASE" "$LOCAL" "$REMOTE" "$MERGED"'

git config --global mergetool.prompt false

git merge GitWork

cat hello.xml

<<<<<<< HEAD

<message>Hello from master branch</message>

=======

<message>Hello from GitWork branch</message>

<note>Conflict demo content</note>

>>>>>>> GitWork

git mergetool

git add hello.xml

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

git status

echo "\*.BACKUP.\*" >> .gitignore

git add .gitignore

git commit -m "Ignore P4Merge backup files"

git branch -a

git branch -d GitWork

git log --oneline --graph --decorate

1. Git-HOL

git checkout master

git status

On branch master

nothing to commit, working tree clean

git branch -a

git pull origin master

git push origin master

git push origin GitWork

git log origin/master --oneline