**1. GIT-HOL**

**Step 1 – Setup Git Configuration**

1. **Check if Git is installed**

bash

CopyEdit

git --version

1. **Set your username and email** (replace with your info)

bash

CopyEdit

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

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

1. **Verify configuration**

bash

CopyEdit

git config --list

**Step 2 – Set Notepad++ as Default Editor**

1. **Check if Notepad++ works from Git Bash**

bash

CopyEdit

notepad++

1. **Create an alias for Notepad++**

bash

CopyEdit

alias np='notepad++'

1. **Set Notepad++ as Git default editor**

bash

CopyEdit

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

1. **Verify default editor**

bash

CopyEdit

git config --global -e

**Step 3 – Create and Track Files in Git**

1. **Create a new project folder**

bash

CopyEdit

mkdir GitDemo

cd GitDemo

1. **Initialize Git repository**

bash

CopyEdit

git init

1. **Create a file**

bash

CopyEdit

echo "Welcome to Git" > welcome.txt

1. **Check status**

bash

CopyEdit

git status

1. **Stage the file**

bash

CopyEdit

git add welcome.txt

1. **Commit the file** (opens Notepad++ for multi-line message)

bash

CopyEdit

git commit

Or commit with a single-line message:

bash

CopyEdit

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

**Step 4 – Push to Remote Repository**

1. **Create a remote repo on GitLab** (named **GitDemo**).
2. **Link local repo to remote**

bash

CopyEdit

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

1. **Pull from remote (if needed)**

bash

CopyEdit

git pull origin master

1. **Push to remote**

bash

CopyEdit

git push origin master

**2. GIT-HOL**

cd GitDemo

**Step 1: Create a .log file and a logs folder**

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

mkdir logs

echo "Inside logs folder" > logs/info.txt

**Step 2: Create .gitignore file**

notepad++ .gitignore

Inside .gitignore file, add:

bash

Copy

Edit

# Ignore all .log files

\*.log

# Ignore logs folder

logs/

bash

Copy

Edit

**# Step 3: Check Git status**

git status

git rm --cached error.log

git rm -r --cached logs/

**Step 4: Add and commit .gitignore**

git add .gitignore

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

**Step 5: Push changes to remote**

git push origin master

**3. GIT-HOL**

### ****Step 1 – Create and Work on a Branch****

bash

CopyEdit

**1. Create a new branch**

git branch GitNewBranch

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

git branch

git branch -a

**3. Switch to the new branch**

git checkout GitNewBranch

**4. Create a new file in this branch**

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

**5. Stage and commit the changes**

git add branchfile.txt

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

**6. Check status**

git status

### ****Step 2 – Merge Back into Master****

bash

CopyEdit

**1. Switch to master**

git checkout master

**2. See command-line differences between master and branch**

git diff GitNewBranch

**3. Set P4Merge as the Git diff tool**

git config --global diff.tool p4merge

git config --global difftool.p4merge.path "C:/Program Files/Perforce/p4merge.exe"

**4. Run P4Merge to see visual differences**

git difftool GitNewBranch

**5. Merge branch into master**

git merge GitNewBranch

**6. View commit history with graph**

git log --oneline --graph --decorate

**7. Delete the branch after merge**

git branch -d GitNewBranch

**8. Check status**

git status

**4. GIT-HOL**

### ****Step 1 – Verify Master is Clean****

bash

CopyEdit

# Check current branch and status

git checkout master

git status

You should see: working tree clean.

### ****Step 2 – Create and Work on**** GitWork ****Branch****

bash

CopyEdit

# Create branch

git branch GitWork

git checkout GitWork

# Add hello.xml with some content

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

# Stage and commit

git add hello.xml

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

### ****Step 3 – Create Different hello.xml in Master****

bash

CopyEdit

# Switch to master

git checkout master

# Add hello.xml with DIFFERENT content

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

# Stage and commit

git add hello.xml

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

### ****Step 4 – View Logs****

bash

CopyEdit

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

### ****Step 5 – Compare Branch & Master****

bash

CopyEdit

# CLI diff

git diff GitWork

# Configure P4Merge as visual diff tool (only once)

git config --global diff.tool p4merge

git config --global difftool.p4merge.path "C:/Program Files/Perforce/p4merge.exe"

# Run P4Merge to see differences

git difftool GitWork

### ****Step 6 – Merge and Resolve Conflict****

bash

CopyEdit

# Attempt to merge GitWork into master

git merge GitWork

### ****Step 7 – Use P4Merge for Conflict Resolution****

bash

CopyEdit

# Configure merge tool (only once)

git config --global merge.tool p4merge

git config --global mergetool.p4merge.path "C:/Program Files/Perforce/p4merge.exe"

# Open 3-way merge tool

git mergetool

### ****Step 8 – Commit Merged Result****

bash

CopyEdit

# Stage merged file

git add hello.xml

# Commit merge

git commit -m "Resolved merge conflict for hello.xml between master and GitWork"

### ****Step 9 – Update**** .gitignore ****to Ignore Backup Files****

bash

CopyEdit

echo "\*~" >> .gitignore # Ignore backup files (~ suffix)

git add .gitignore

git commit -m "Updated .gitignore to ignore backup files"

### ****Step 10 – Clean Up****

bash

CopyEdit

# List all branches

git branch

# Delete merged branch

git branch -d GitWork

# View final log

git log --oneline --graph --decorate

**5. GIT-HOL**

**Step 1 – Verify if master is clean**

git checkout master

git status

**Step 2 – List all available branches**

git branch -a

**Step 3 – Pull remote repository updates into master**

git pull origin master

**Step 4 – Push pending changes from Git-T03-HOL\_002 to remote**

git push origin master