**Digital Nurture 4.0**

**Week 8 - GIT**

# **Mandatory HandsOn**

# ***1.Git-HOL***

**Step1 : Check if Git is installed**

bash

CopyEdit

git –version

**Step 2: Configure Git username & email**

bash

CopyEdit

git config --global user.name "Abulhasan06"

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

Verify:

bash

CopyEdit

git config --list

**Step 3 : Set Git editor**

**Option A — Windows Notepad**

bash

CopyEdit

git config --global core.editor "notepad"

**Option B — Notepad++**

* Install Notepad++
* Add C:\Program Files\Notepad++ to PATH (Environment Variables → User Path → New)
* Test:

bash

CopyEdit

notepad++

* Set as Git editor:

bash

CopyEdit

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

**Step 4 : Create Git repository**

bash

CopyEdit

cd ~

mkdir GitDemo

cd GitDemo

git init

**Step 5 : Create a file**

bash

CopyEdit

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

ls

cat welcome.txt

**Step 6 : Stage & commit**

bash

CopyEdit

git add welcome.txt

git commit -m "Added welcome.txt file for demo"

Verify:

bash

CopyEdit

git status

git log --oneline

**Step 7 : Add remote & push to GitHub**

bash

CopyEdit

git remote add origin https://github.com/Abulhasan06/GitDemo.git

git remote -v

git push -u origin master

**Step 8 : Pull from remote**

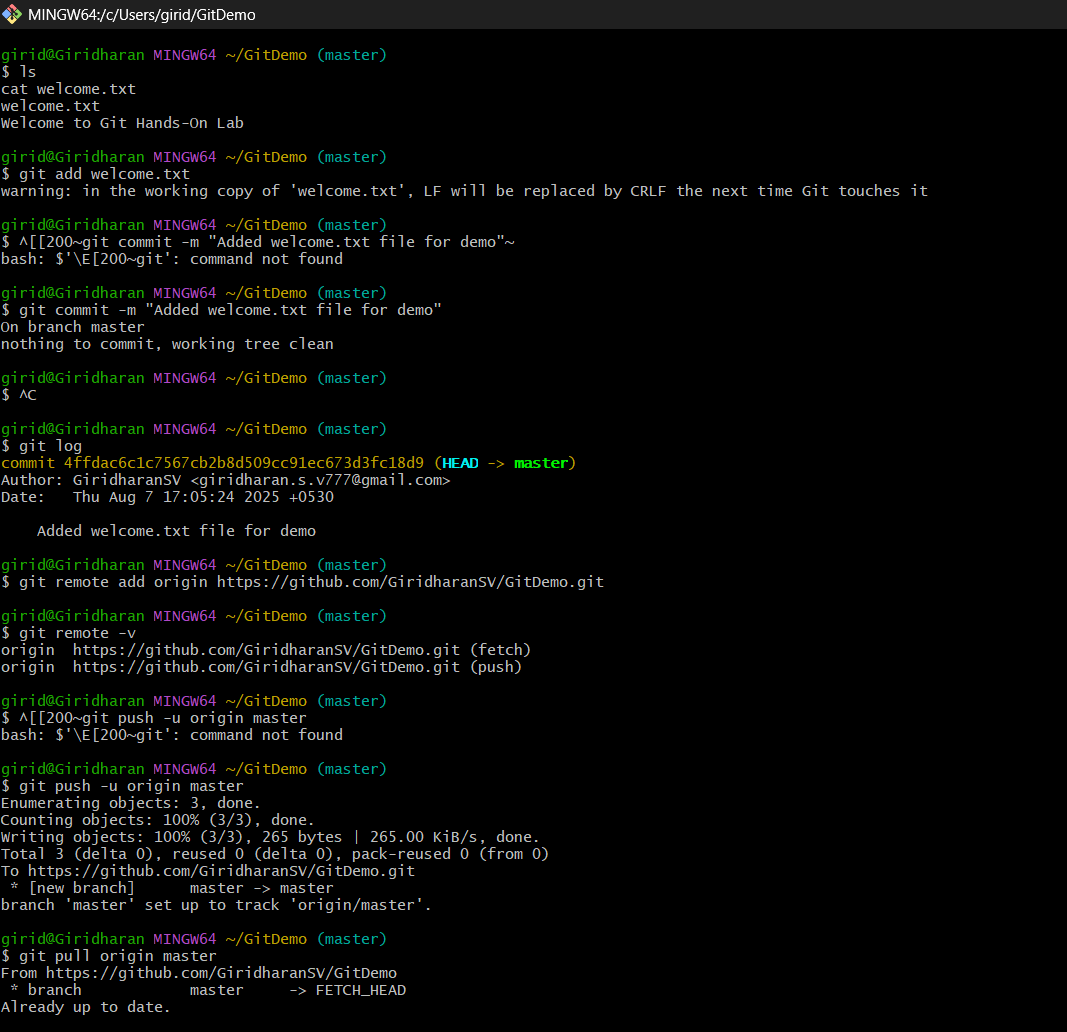
bash

CopyEdit

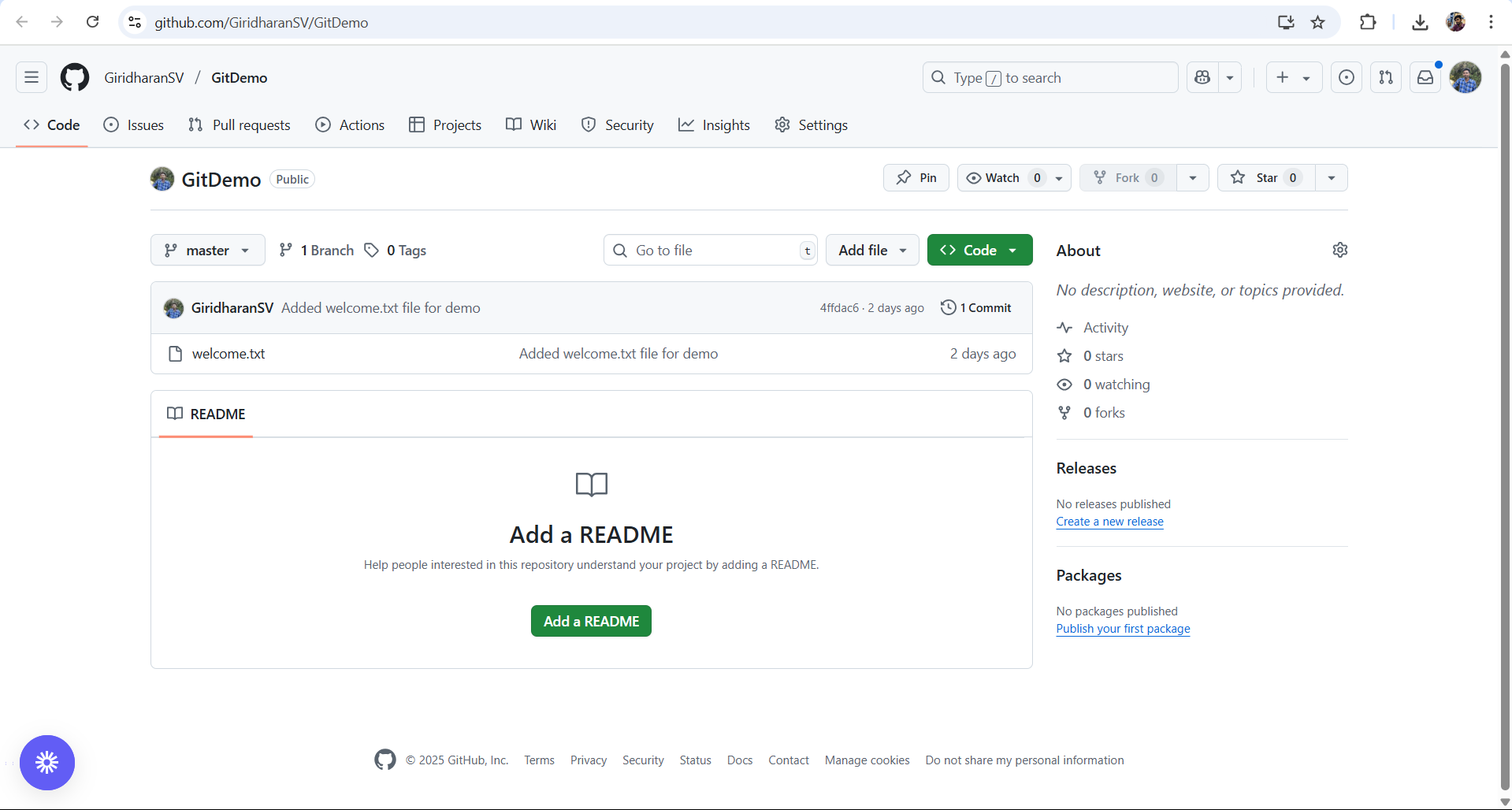
git pull origin master

**Output:**

In GitBash:



**In GitHub:**



A close-up of a person's hand

AI-generated content may be incorrect.

# ***2.Git-HOL***

**Step 1 : Create sample .log file & folder**

bash

CopyEdit

echo "error log content" > error.log

mkdir log

echo "inner log content" > log/inner.log

**Step 2 : Create .gitignore file**

bash

CopyEdit

printf "%s\n" "# ignore log files and log folder" "\*.log" "log/" > .gitignore

**.gitignore** content should be:

bash

CopyEdit

# ignore log files and log folder

\*.log

log/

**Step 3 : Verify .gitignore is ignoring files**

bash

CopyEdit

git status

git status --ignored

.gitignore appears as **untracked**  
.log files & log/ folder appear in **Ignored files** section

**Step 4 : Stage & commit .gitignore**

bash

CopyEdit

git add .gitignore

git commit -m "Add .gitignore to ignore \*.log and log/ folder"

**Step 5 : Push changes to GitHub**

bash

CopyEdit

git push

**Step 6 : Verify on GitHub**

* Open the repo → .gitignore file should be visible
* .log files should **not** be in the repository

**Output :**

**In GitBash:**

A screenshot of a computer

AI-generated content may be incorrect.

**In GitHub:**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

# ***3.Git-HOL***

**Step 1 : Create new branch**

**Command:**

css

Copy code

git branch GitNewBranch

git branch -a

**Explanation:** Create a new branch named GitNewBranch and list all branches.

**Step 2 : Switch to new branch**

**Command:**

nginx

Copy code

git checkout GitNewBranch

git branch

**Step 3 : Add file in branch**

**Command:**

bash

Copy code

echo "This is GitNewBranch content" > branchfile.txt

ls

cat branchfile.txt

**Step 4 : Stage & commit changes**

**Command:**

sql

Copy code

git add branchfile.txt

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

**Step 5 : Switch back to master**

**Command:**

bash

Copy code

git checkout master

ls

**Step 6 : Merge branch into master**

**Command:**

sql

Copy code

git merge GitNewBranch

**Step 7 : View commit history**

**Command:**

css

Copy code

git log --oneline --graph --decorate

**Step 8 : Delete branch**

**Command:**

nginx

Copy code

git branch -d GitNewBranch

git branch

**Step 9 : Push changes to GitHub**

**Command:**

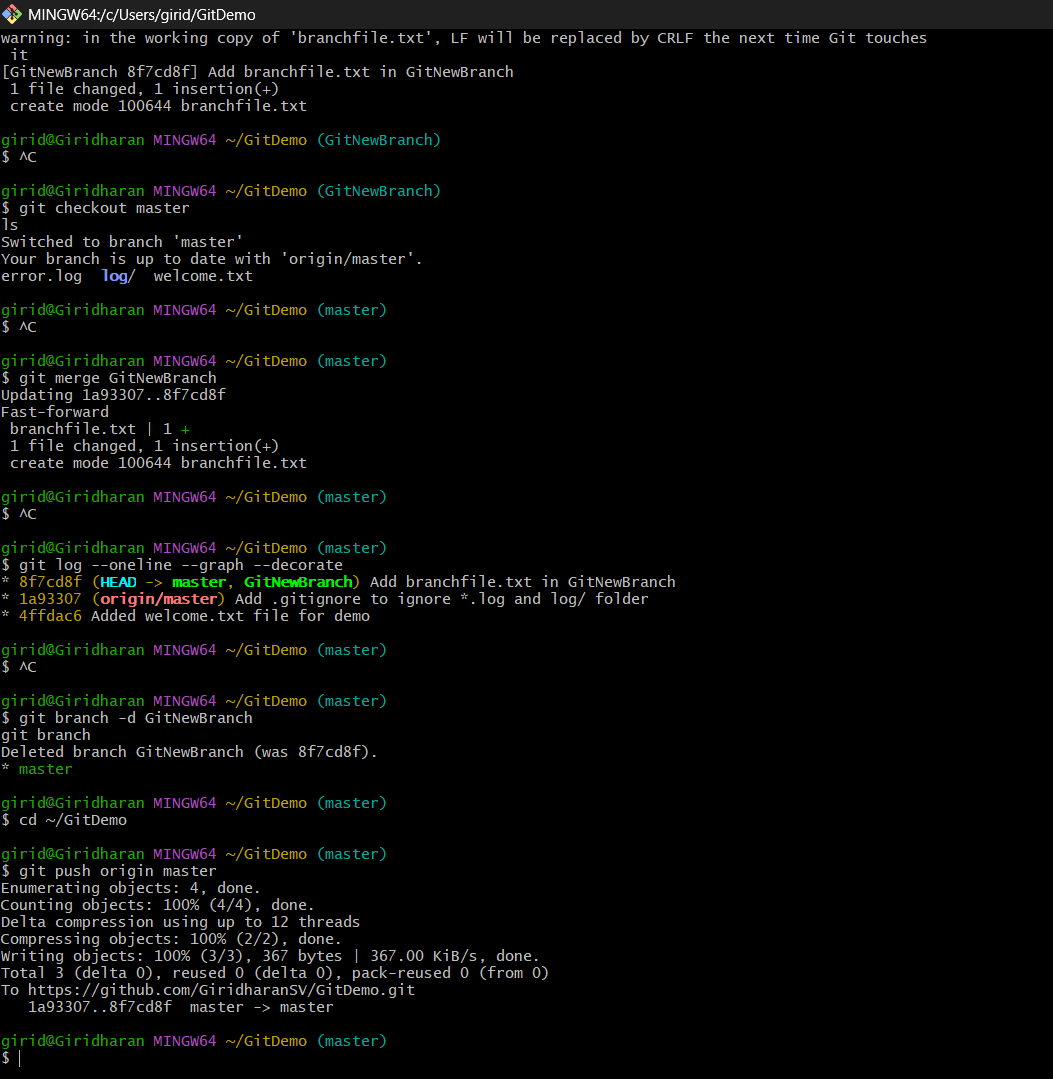
perl

Copy code

git push origin master

**Output:**

In GitBash:



In Github:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

# ***4.Git-HOL***

**Step 1 : Check repository location and status**

bash

Copy code

pwd

git status

git branch -a

**Step 2 : Create a new branch**

bash

Copy code

git checkout -b GitWork

**Step 3 : Create a file and commit it (in GitWork branch)**

bash

Copy code

notepad hello.xml

Add the following content:

xml

Copy code

<hello>Hi from GitWork branch</hello>

Save and close the file.

bash

Copy code

git add hello.xml

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

git status

**Step 4 : Switch to master and create a different version of the file**

bash

Copy code

git checkout master

notepad hello.xml

Add the following content:

xml

Copy code

<hello>Hello from master branch (different content)</hello>

Save and close the file.

bash

Copy code

git add hello.xml

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

**Step 5 : Merge and create a conflict**

bash

Copy code

git merge GitWork

If Git performs a fast-forward merge (no conflict), force a conflict by:

bash

Copy code

git reset --hard HEAD~1

# Recreate master version with different content, add & commit

git merge GitWork

**Step 6 : Resolve conflict using P4Merge**

Configure P4Merge (only once):

bash

Copy code

git config --global merge.tool p4merge

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

Run the merge tool:

bash

Copy code

git mergetool

Choose the final merged content in the output pane, save, and close P4Merge.

**Step 7 : Commit the resolved merge**

bash

Copy code

git add hello.xml

git commit -m "Merge GitWork into master — resolved with P4Merge"

**Step 8 : Ignore backup files**

bash

Copy code

cp hello.xml hello.xml.bak

echo "hello.xml.bak" >> .gitignore

git add .gitignore

git commit -m "Ignore backup files"

**Step 9 : Delete the merged branch**

bash

Copy code

git branch -d GitWork

**Step 10 : Check the final commit history**

bash

Copy code

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

**Step 11 : Push changes to GitHub**

bash

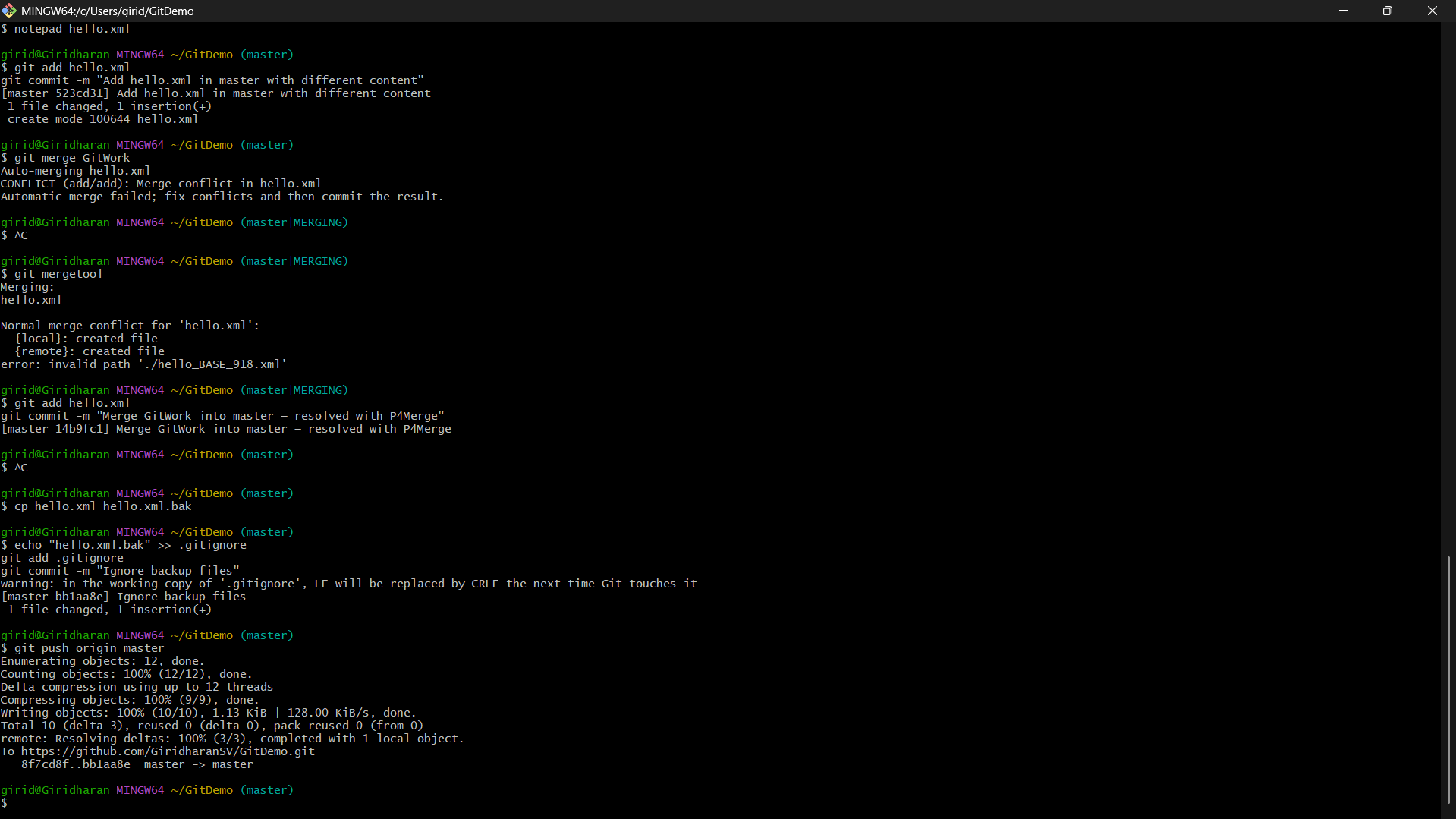
Copy code

git remote -v

git push origin master

**Output:**

In GitBash:



In Github:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

# ***5.Git-HOL***

**Step 1 : Navigate to the Project Folder**

bash

CopyEdit

cd path/to/GitDemo

pwd

ls -la

**Step 2 : Check Repository Status**

bash

CopyEdit

git status

git branch -a

**Step 3 : Update the Master Branch**

bash

CopyEdit

git checkout master

git pull origin master

**Step 4 : Stage and Commit Untracked Files**

bash

CopyEdit

git add hello.xml.orig

git commit -m "Add hello.xml.orig"

**Step 5 : Create a New Branch**

bash

CopyEdit

git checkout -b Git-T03-HOL\_002

**Step 6 : Make Changes and Commit**

bash

CopyEdit

git add -A

git commit -m "Completed Git HOL task"

**Step 7 : Push the Branch to Remote**

bash

CopyEdit

git push -u origin Git-T03-HOL\_002

**Step 8 : Create and Merge Pull Request (on GitHub)**

On GitHub, open the pull request link generated after the push, choose master as the base branch and Git-T03-HOL\_002 as the compare branch, give a title and description, click **Create Pull Request**, review the changes, and merge.

**Step 9 : Update Master Branch After Merge**

Switch back to the master branch and pull the latest merged changes from the remote repository.

bash

CopyEdit

git checkout master

git pull origin master

**Step 10 – Delete the Feature Branch**

Delete the merged branch both locally and on the remote repository.

bash

CopyEdit

git branch -d Git-T03-HOL\_002

git push origin --delete Git-T03-HOL\_002

**Output:**

In GitBash:

A screen shot of a computer

AI-generated content may be incorrect.

In Github:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.