**1.GIT – HOL**

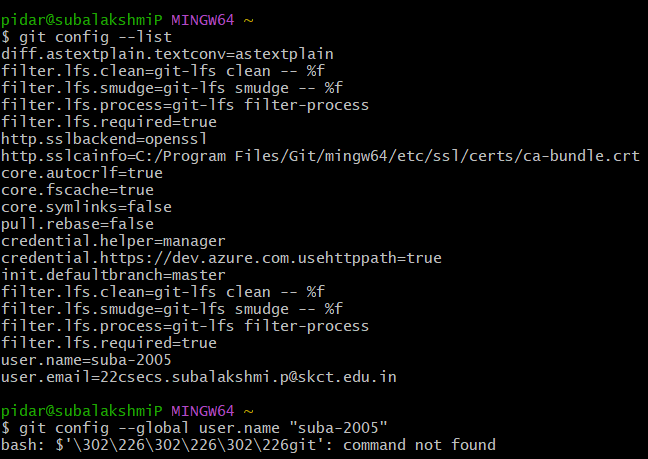
**Git Configuration**

**Commands used:**

git config --global user.name "Janani K"

git config --global user.email "22csecs.janani.k@skct.edu.in"

**Output (example):**



**Integrating Notepad++**

**Steps:**

1. Locate Notepad++ (e.g., C:\Program Files\Notepad++)
2. Add to PATH via:
   * Control Panel → System → Advanced system settings → Environment Variables
   * Edit "Path" under User Variables → Add Notepad++ path
3. Set as default Git editor: git config --global core.editor "notepad++"
4. Verify: git config --global -e

**6. Create Local Repository**

mkdir GitDemo

cd GitDemo

git init

**7. Add File to Repository**

echo "Welcome to GitLab hands-on session" > welcome.txt

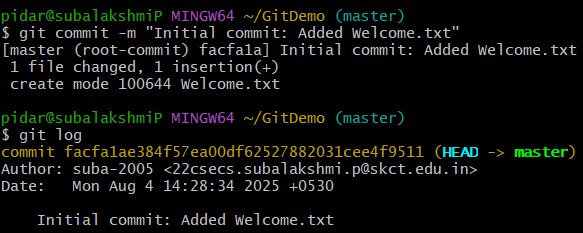
git add welcome.txt

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

**8. Verify Commit**

git log

**OUTPUT**



**2. GIT – HOL**

**1. Create a remote GitLab repository (if not already done)**

* Go to <https://gitlab.com/>
* Click **➕ New Project**
* Name it: GitDemo
* Visibility: **Private** or **Public** (your choice)
* Click **Create Project**

**2. Copy the HTTPS clone URL**

https://gitlab.com/your-username/GitDemo.git

**3. Link your local repo to GitLab**

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

(Replace the URL with your actual GitLab repo link)

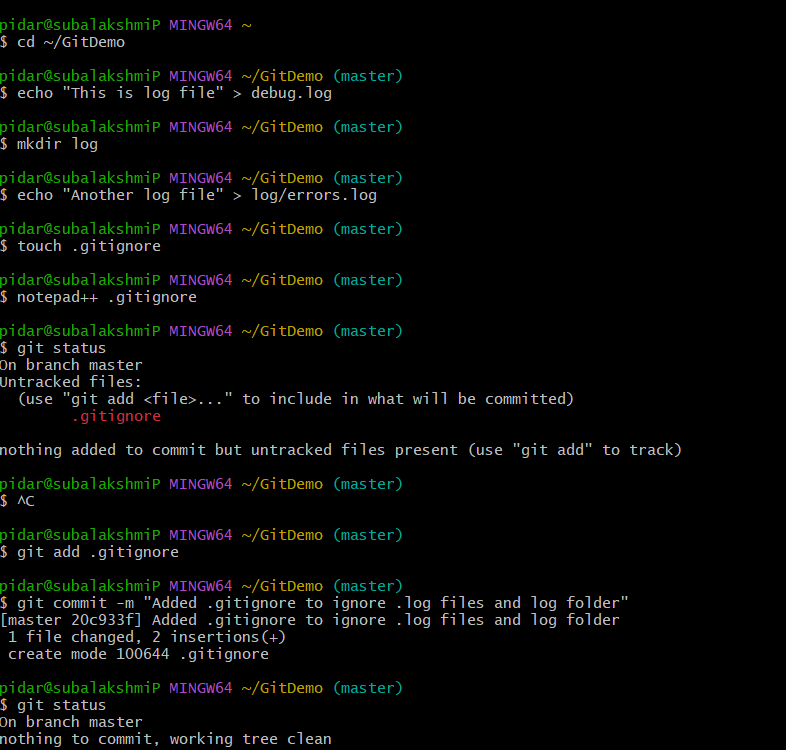
**4. Push your code to GitLab**

If this is your **first push**, use:

git push -u origin master

(If your default branch is main on GitLab, replace master with main)

**OUTPUT**



**3.GIT – HOL**

## ****PART 1: Branching in Git****

### Step 1: Create and switch to a new branch

git checkout -b GitNewBranch

### Step 2: List all branches

git branch -a

### Step 3: Add a file and write content

echo "This is a test file in GitNewBranch" > newfile.txt

### Step 4: Stage the file

git add newfile.txt

### Step 5: Commit the changes

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

### Step 6: Check status

git status

## ****PART 2: Merging in Git****

### Step 1: Switch to master

git checkout master

### Step 2: Show CLI differences

git diff master GitNewBranch

### Step 3: Visual difference using P4Merge

git difftool master GitNewBranch

P4Merge opens and shows visual diff.

### Step 4: Merge GitNewBranch into master

git merge GitNewBranch

### Step 5: Visualize merge history

git log --oneline --graph --decorate

### Step 6: Delete the merged branch

git branch -d GitNewBranch

## ****PART 3: Using GitLab for Branch and Merge Requests****

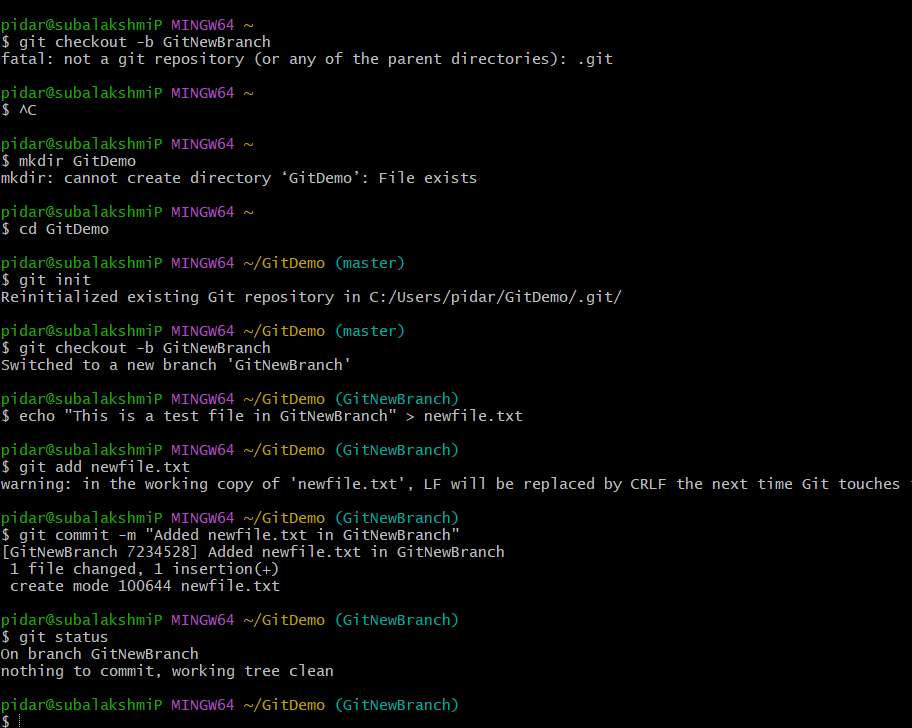
### Create a Branch in GitLab:

1. Go to your GitLab project → Repository → Branches
2. Click on **New branch**
3. Name it GitNewBranch, and base it off main
4. Click **Create branch**

### Push your local branch to GitLab:

git push origin GitNewBranch

**OUTPUT**



**4.GIT – HOL**

### Step 1: Initialize a Git repository (if not already done)

mkdir GitLabConflictDemo

cd GitLabConflictDemo

git init

### Step 2: Create a base file and make an initial commit on master

echo "Initial setup" > setup.txt

git add setup.txt

git commit -m "Initial commit on master with setup.txt"

### Step 3: Create a new branch GitWork and switch to it

git checkout -b GitWork

### Step 4: Create hello.xml and add content in GitWork

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

git add hello.xml

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

### Step 5: Switch back to master

git checkout master

### Step 6: Create a different version of hello.xml in master

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

git add hello.xml

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

### Step 7: View log history of both branches

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

### Step 8: Check differences via CLI

git diff GitWork master

### Step 9: View differences visually using P4Merge

git difftool GitWork master

### Step 10: Merge GitWork into master

git merge GitWork

### Step 11: Observe the conflict markers in hello.xml

cat hello.xml

### Step 12: Resolve conflict using 3-way merge tool (P4Merge)

git mergetool

### Step 13: Add and commit the resolved file

git add hello.xml

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

### Step 14: Add .orig backup file pattern to .gitignore

echo "\*.orig" >> .gitignore

git add .gitignore

git commit -m "Ignore merge backup files"

### Step 15: List all branches

### git branch

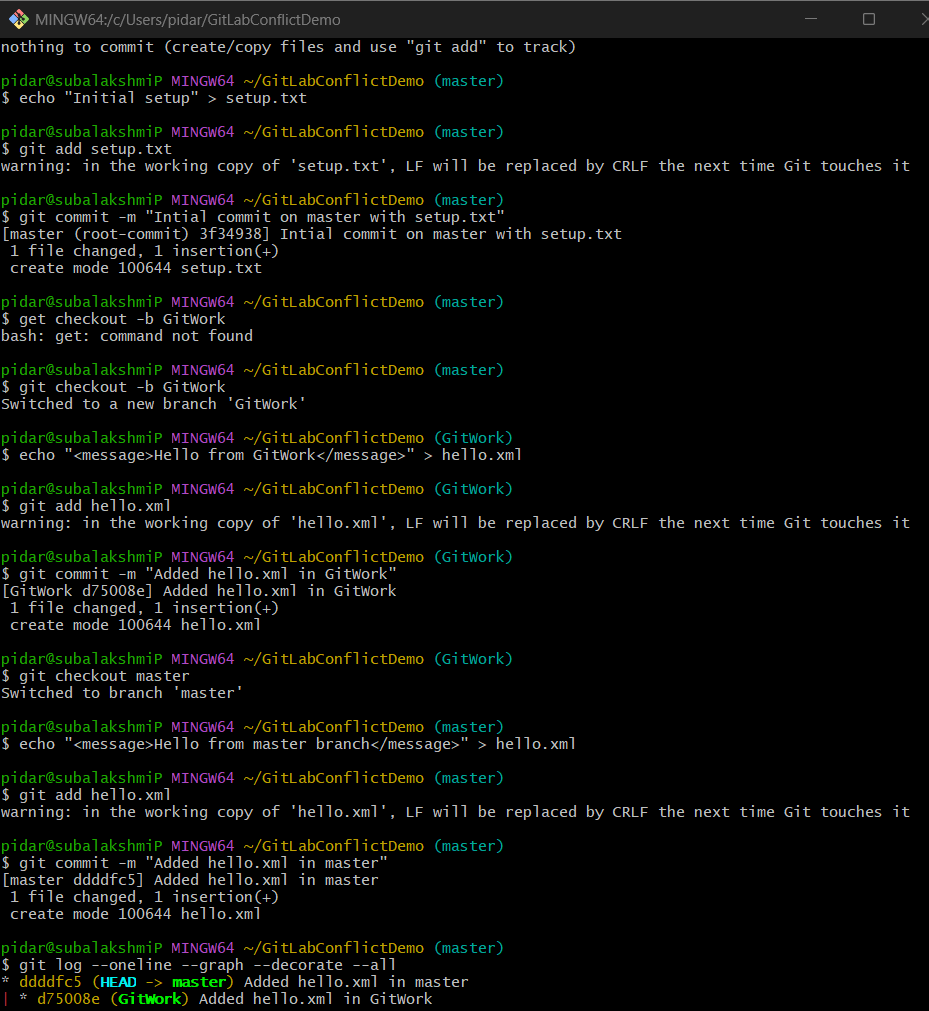
### Step 16: Delete the merged branch

git branch -d GitWork

### Step 17: View final log history

git log --oneline --graph --decorate

**OUTPUT**



**5. GIT – HOL**

**Step 1: Create a new Git project folder and initialize Git**

mkdir GitPushDemo

cd GitPushDemo

git init

**Step 2: Create and commit a sample file**

echo "This is from Git-T03-HOL\_002" > lab\_notes.txt

git add lab\_notes.txt

git commit -m "Added lab\_notes.txt as part of Git-T03-HOL\_002"

**Step 3: Create a new repository on GitHub**

1. Go to <https://github.com>
2. Click **New repository**
3. Enter repository name: GitPushDemo
4. Choose public or private
5. **Do not** initialize with README
6. Click **Create repository**

**Step 4: Link local repo with remote GitHub repo**

Replace <your-username> with your GitHub username:

git remote add origin https://github.com/<your-username>/GitPushDemo.git

**Step 5: Verify if master is in clean state**

git status

**Step 6: List available branches**

git branch -a

**Step 7: Pull from remote (first time repo might not have content)**

git pull origin master

**Step 8: Push local commits to remote GitHub repo**

git push -u origin master

**Step 9: Verify the changes in the remote repo**

1. Go to https://github.com/<your-username>/GitPushDemo
2. You should see:
   * lab\_notes.txt
   * Your commit message: **“Added lab\_notes.txt as part of Git-T03-HOL\_002”**

**OUTPUT**

