1.GIT-HOL

GIT SETUP GUIDE

Git is a powerful version control system used to track changes in code and collaborate with others. In this lab, you'll learn to install and configure Git, set Notepad++ as the default editor, and perform basic Git operations like commit, push, and pull using Git Bash.

Objectives:

- Learn and practice basic Git commands: git init, git status, git add, git commit, git push, git pull
- Configure Git on your machine with your username and email
- Set Notepad++ as the default Git editor
- Create and track a file in a Git repository
- Push changes to a remote repository on GitLab

Requirements:

- Install Git Bash client
- Install Notepad++
- Create a free GitLab account (Do not use Cognizant credentials)

Step 1: Setup Git Configuration

1. Verify Git Installation:
```
git --version
```
2. Configure User Information:
```
git config --global user.name "Your Name"
git config --global user.email "your@email.com"
```
3. Verify Configuration:
```
git config --list
```

Step 2: Integrate Notepad++ as Default Editor

1. Check if Notepad++ works:
```
notepad++
```

2. If not recognized, add path of notepad++.exe to Environment Variables

3. Create an Alias:
   ```
   alias np='notepad++'
   echo "alias np='notepad++'">> ~/.bashrc
   ```

4. Set Notepad++ as Git Default Editor:
   ```
   git config --global core.editor "notepad++ -multiInst -nosession"
   ```

5. Verify Editor Configuration:
   ```
   git config --global -e
   ```

Step 3: Create and Manage a Repository

1. Create a New Project:
   ```
   mkdir GitDemo
   cd GitDemo
   git init
   ```

2. Check Initialization:
   ```
   ls -a
   ```

3. Create a File:
   ```
   echo "Welcome to Git Demo!"> welcome.txt
   ```

4. Verify File:
   ```
   ls
   cat welcome.txt
   ```

5. Check Git Status:
   ```
   git status
   ```

6. Add File to Git Tracking:
   ```
   git add welcome.txt
   ```

7. Commit the File:
   git commit -m "Added welcome.txt file"or use git commit for multi-line message

8. Verify Status:
   ```
   git status
   ```

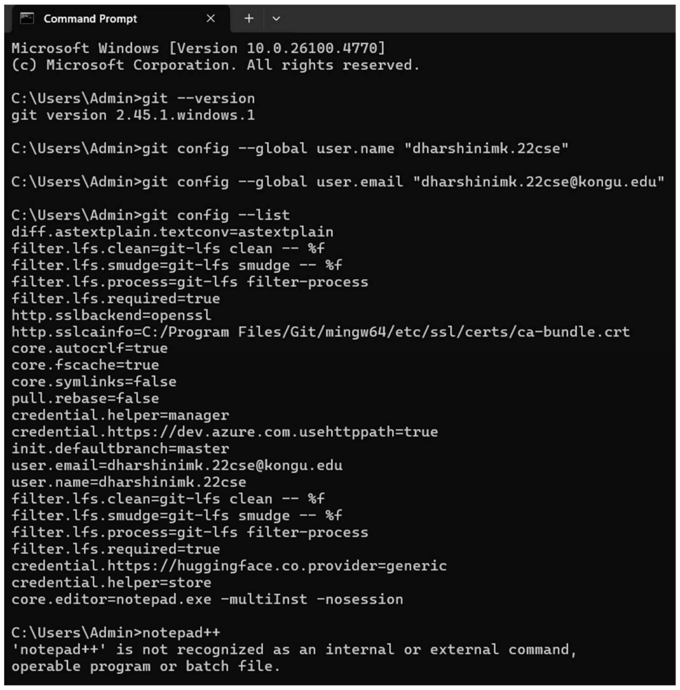
Step 4: Connect to Remote Repository

1. Create a Remote Repository on GitLab (Project: GitDemo)
2. Link Remote Repo:
```
git remote add origin <your-gitlab-repo-url>
```
3. Pull Remote Changes:
```
git pull origin master
```
4. Push Local Repo to Remote:
```
git push origin master
```

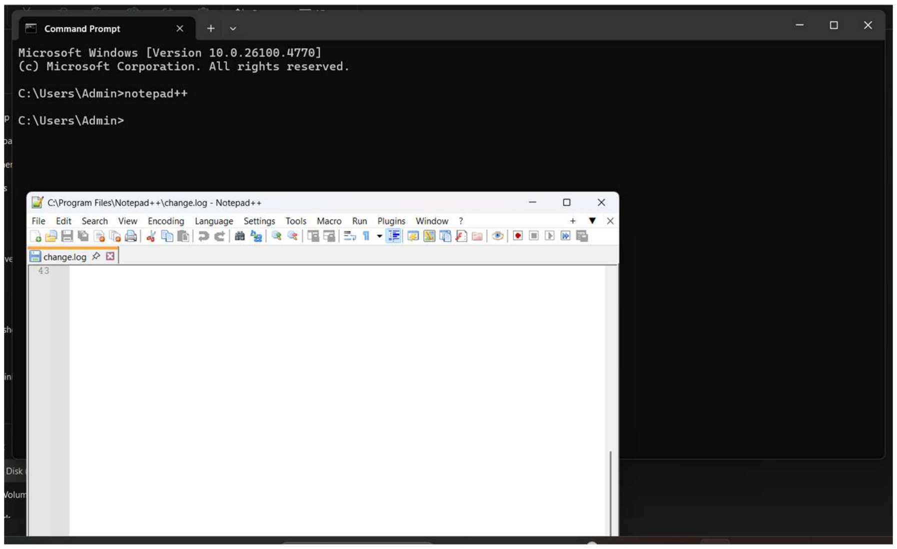
Git Commands Cheat Sheet

Command	Purpose
git init	Initialize a new repository
git status	Show current status
git add <file>	Stage a file for commit
git commit -m "msg"	Commit with a message
git pull origin master	Fetch & merge from remote
git push origin master	Push local changes to remote

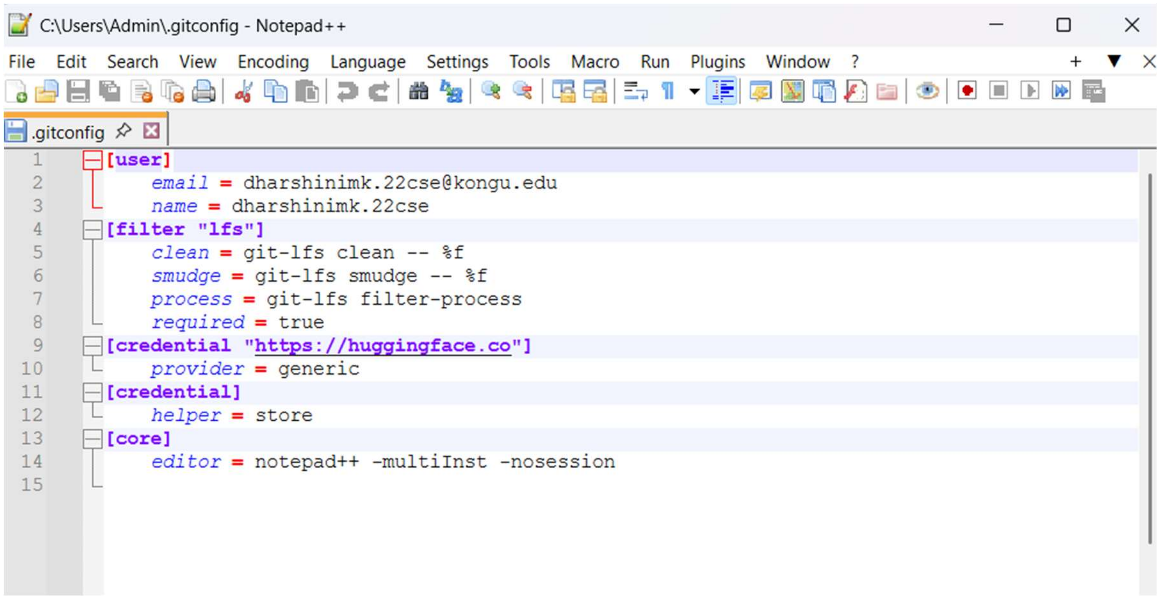
Output:



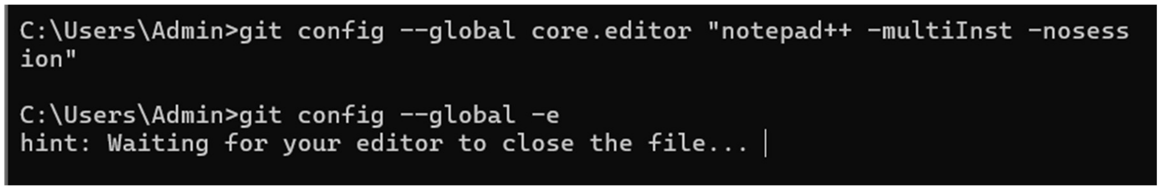
```
Microsoft Windows [Version 10.0.26100.4770]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>git --version
git version 2.45.1.windows.1

C:\Users\Admin>git config --global user.name "dharshinimk.22cse"

C:\Users\Admin>git config --global user.email "dharshinimk.22cse@kongu.edu"

C:\Users\Admin>git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/etc/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
user.email=dharshinimk.22cse@kongu.edu
user.name=dharshinimk.22cse
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
credential.https://huggingface.co.provider=generic
credential.helper=store
core.editor=notepad.exe -multiInst -nosession

C:\Users\Admin>notepad++
'notepad++' is not recognized as an internal or external command,
operable program or batch file.
```



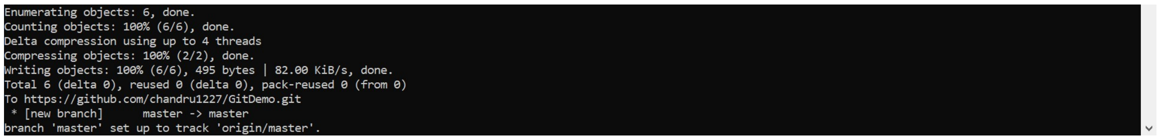
```
Microsoft Windows [Version 10.0.26100.4770]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>notepad++

C:\Users\Admin>
```


C:\Users\Admin\.gitconfig - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
.gitconfig

```
[user]
	email = dharshinimk.22cse@kongu.edu
	name = dharshinimk.22cse
[filter "lfs"]
	clean = git-lfs clean -- %f
	smudge = git-lfs smudge -- %f
	process = git-lfs filter-process
	required = true
[credential "https://huggingface.co"]
	provider = generic
[credential]
	helper = store
[core]
	editor = notepad++ -multiInst -nosession
```



```
C:\Users\Admin>git config --global core.editor "notepad++ -multiInst -nosess
ion"

C:\Users\Admin>git config --global -e
hint: Waiting for your editor to close the file...
```



```
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (6/6), 495 bytes | 82.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/chandru1227/GitDemo.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```

2. GIT-HOL

GIT IGNORE COMMAND

When working on a project, certain files like logs, temporary files, and build outputs don't need to be tracked by Git. Using .gitignore, we can tell Git to ignore these unnecessary files and folders. This helps keep the repository clean and avoids pushing unwanted data to remote repositories.

Objectives:

·Understand the purpose of .gitignore

·Learn how to ignore specific files and folders using .gitignore

·Apply the .gitignore rule to ignore .log files and log/ folders

·Confirm that ignored files are not tracked using git status.

Prerequisites

- Git environment setup
- Integration of Notepad++ as the default Git editor
- A Git repository available in the local system
- A remote repository on GitHub or GitLab
- A free GitHub account (Do not use Cognizant credentials)

Step 1: Create a GitHub Account

1. Go to https://github.com.
2. Sign up for a free account using your personal email (not Cognizant credentials).
3. Verify your email and log in.

Step 2: Set Up Local Repository

1. Open Git Bash or Command Prompt.
2. Navigate to your working directory:
```
cd path/to/your/project
```
3. Initialize Git if not already done:
```
git init
```

Step 3: Create .log File and log Folder

```
echo "This is a log file"> test.log
mkdir log
echo "Folder log created"> log/info.txt
```

Step 4: Configure .gitignore

1. Create a .gitignore file in the root of your repository:
```
notepad++ .gitignore
```
2. Add the following lines to ignore .log files and the log folder:
```
*.log
log/
```

Step 5: Verify .gitignore

Run:

```
git status
```

You should not see test.log or the log folder in the tracked changes.

Example output:

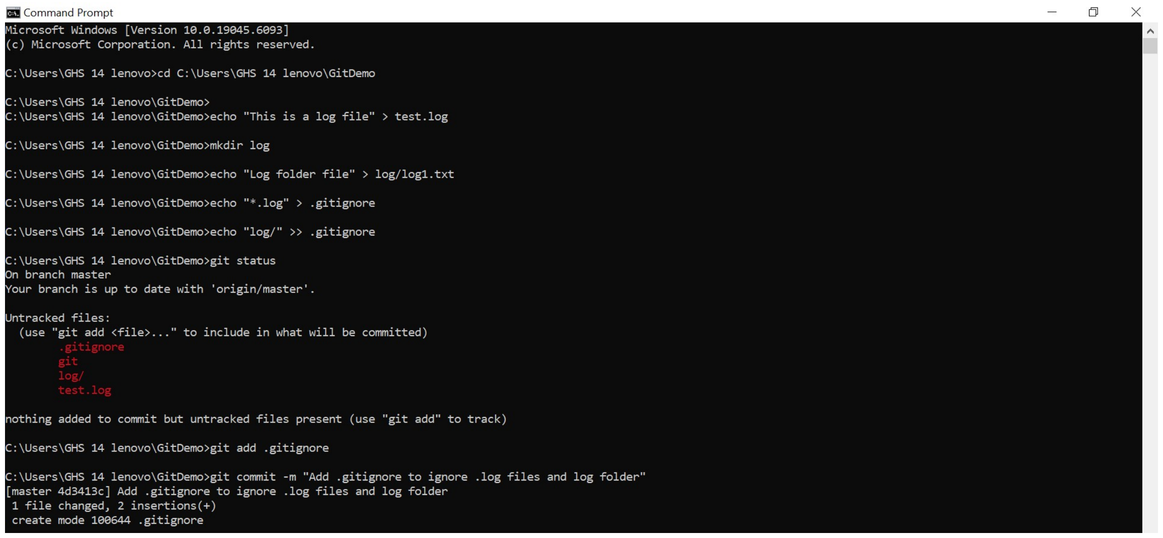
```
On branch main

No commits yet
nothing to commit (create/copy files and use "git add" to track)
```

Step 6: Commit Other Files

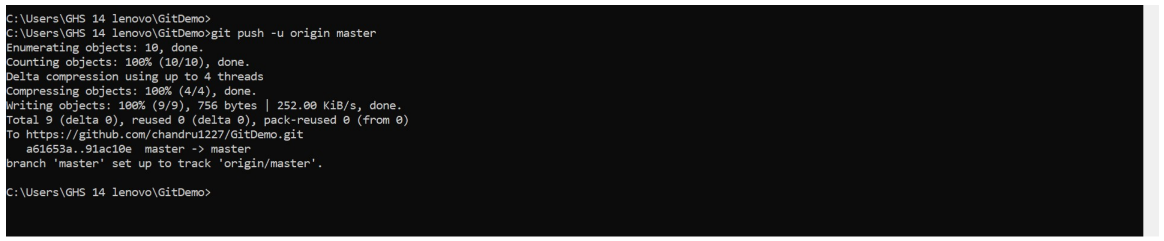
1. Add all non-ignored files:
   ```
   git add .
   ```
2. Commit changes:
   ```
   git commit -m "Added .gitignore and project setup"
   ```

Verification

- .log files and the log folder remain untracked.
- Other project files are committed successfully.



```
Microsoft Windows [Version 10.0.19045.6093]
(c) Microsoft Corporation. All rights reserved.

C:\Users\GHS 14 lenovo>cd C:\Users\GHS 14 lenovo\GitDemo

C:\Users\GHS 14 lenovo\GitDemo>
C:\Users\GHS 14 lenovo\GitDemo>echo "This is a log file" > test.log

C:\Users\GHS 14 lenovo\GitDemo>mkdir log

C:\Users\GHS 14 lenovo\GitDemo>echo "Log folder file" > log/log1.txt

C:\Users\GHS 14 lenovo\GitDemo>echo "*.log" > .gitignore

C:\Users\GHS 14 lenovo\GitDemo>echo "log/" >> .gitignore

C:\Users\GHS 14 lenovo\GitDemo>git status
On branch master
Your branch is up to date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        git
        log/
        test.log

nothing added to commit but untracked files present (use "git add" to track)

C:\Users\GHS 14 lenovo\GitDemo>git add .gitignore

C:\Users\GHS 14 lenovo\GitDemo>git commit -m "Add .gitignore to ignore .log files and log folder"
[master 4d3413c] Add .gitignore to ignore .log files and log folder
 1 file changed, 2 insertions(+)
 create mode 100644 .gitignore
```



```
C:\Users\GHS 14 lenovo\GitDemo>
C:\Users\GHS 14 lenovo\GitDemo>git push -u origin master
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (9/9), 756 bytes | 252.00 KiB/s, done.
Total 9 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/chandru1227/GitDemo.git
   a61653a..91ac10e  master -> master
branch 'master' set up to track 'origin/master'.

C:\Users\GHS 14 lenovo\GitDemo>
```

3.GIT-HOL

GIT Branching and Merging

In software development, **branching and merging** are essential for managing code changes efficiently. Branches allow developers to work on features or fixes in isolation, while merging brings these changes back into the main project. GitLab simplifies this process through branch and merge request features, enabling smooth collaboration in teams.

Objectives:

·Understand **branching and merging** concepts in Git.

·Learn to **create a branch request** in GitLab.

·Learn to **create a merge request** in GitLab.

·Practice creating a branch, making changes, and **merging it with the master branch**.

Requirements:

Setting up Git environment with P4Merge tool for Windows

A GitHub account (create a free account; do **not** use Cognizant credentials)

Part 1: Branching

1. Create a new branch “GitNewBranch”

2. `git checkout -b GitNewBranch`

3. List all local and remote branches

4. `git branch -a`

5. Observe the * mark which denotes the current active branch.

6. Switch to the newly created branch

7. git checkout GitNewBranch

8. Add files with content

9. echo"This is a new file for GitNewBranch"> newfile.txt
   git add newfile.txt

10. Commit the changes

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

12. Check the status

13. git status

Part 2: Merging

1. Switch back to master

2. `git checkout master`

3. List differences between trunk (master) and branch

4. `git diff GitNewBranch`

5. List visual differences using P4Merge tool

6. `git mergetool`

7. Ensure P4Merge is configured as the default mergetool.

8. Merge the source branch into master

9. `git merge GitNewBranch`

10. View log after merging

11. `git log --oneline--graph--decorate`

12. Delete the merged branch

13. `git branch -d GitNewBranch`

14. `git status`

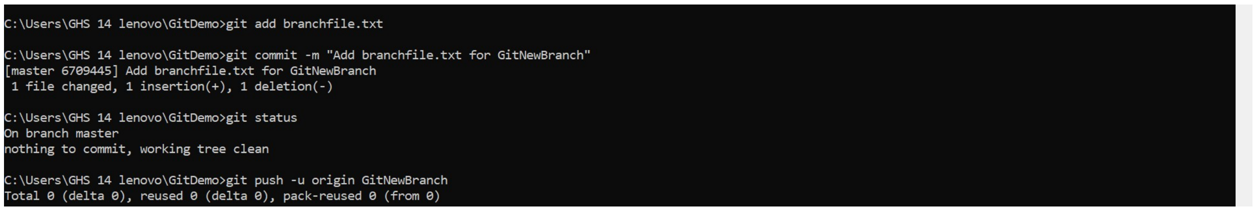
Part 3:Creating Branch and Merge Requests in GitLab

1. Creating a Branch Request
2. Navigate to your GitLab repository.
3. Go to **Repository** > **Branches**.
4. Click **New branch**.
5. Provide branch name (e.g., GitNewBranch) and source branch (master).
6. Click **Create branch**.
7. Creating a Merge Request
8. After pushing your branch, go to **Merge Requests**.
9. Click **New Merge Request**.
10. Select source branch (GitNewBranch) and target branch (master).
11. Add a title and description for your changes.
12. Click **Submit Merge Request**.

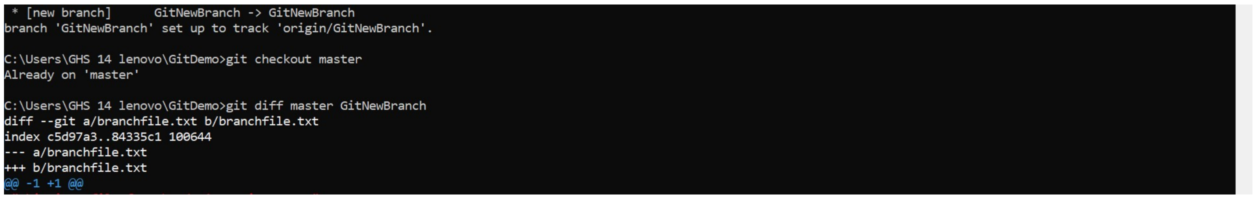
Notes:

1. Always sync (git pull) before creating a new branch.
2. Use clear commit messages for traceability.
3. Ensure all conflicts are resolved before merging.

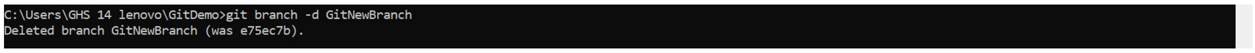
Output:



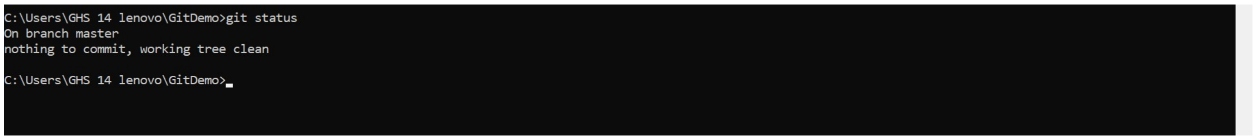
```
C:\Users\GHS 14 lenovo\GitDemo>git add branchfile.txt

C:\Users\GHS 14 lenovo\GitDemo>git commit -m "Add branchfile.txt for GitNewBranch"
[master 6709445] Add branchfile.txt for GitNewBranch
 1 file changed, 1 insertion(+), 1 deletion(-)

C:\Users\GHS 14 lenovo\GitDemo>git status
On branch master
nothing to commit, working tree clean

C:\Users\GHS 14 lenovo\GitDemo>git push -u origin GitNewBranch
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
```



```
 * [new branch]      GitNewBranch -> GitNewBranch
branch 'GitNewBranch' set up to track 'origin/GitNewBranch'.

C:\Users\GHS 14 lenovo\GitDemo>git checkout master
Already on 'master'

C:\Users\GHS 14 lenovo\GitDemo>git diff master GitNewBranch
diff --git a/branchfile.txt b/branchfile.txt
index c5d97a3..84335c1 100644
--- a/branchfile.txt
+++ b/branchfile.txt
@@ -1 +1 @@
```



```
C:\Users\GHS 14 lenovo\GitDemo>git branch -d GitNewBranch
Deleted branch GitNewBranch (was e75ec7b).
```



```
C:\Users\GHS 14 lenovo\GitDemo>git status
On branch master
nothing to commit, working tree clean

C:\Users\GHS 14 lenovo\GitDemo>
```

4.GIT-HOL

GIT Merge Conflicts

In collaborative software development, multiple developers often work on the same codebase simultaneously. This can lead to *merge conflicts* when changes from different branches overlap. Git provides tools and workflows to resolve such conflicts effectively, ensuring code integrity and smooth collaboration.

Objectives:

1. Understand how merge conflicts occur in Git.
2. Learn how to handle and resolve merge conflicts during integration.

3. Practice using tools like Git Bash and P4Merge for conflict visualization and resolution.
4. Gain hands-on experience with:

   Branching and merging

   Conflict detection

   3-way conflict resolution

   Cleaning up post-merge state (e.g., updating `.gitignore`, deleting merged branches)

Requirements:

A GitHub account (please create a free account; do **not** use Cognizant credentials)

Git installed on your system

P4Merge tool installed for visual diff and merge

Step 1: Verify if master is in clean state

```
git checkout master
git status
```

Ensure there are no uncommitted changes.

Step 2: Create a branch “GitWork” and add a file “hello.xml”

```
git checkout -b GitWork
    echo"<greeting>Hello from GitWork branch</greeting>"> hello.xml
    git add hello.xml
    git commit -m"Add hello.xml in GitWork branch"
```

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

```
Echo"<note>This is an update in GitWork branch</note>">> hello.xml
    git status
```

Step 4: Commit the changes to reflect in the branch

```
git add hello.xml
git commit -m"Update hello.xml in GitWork branch"
```

Step 5: Switch to master

```
git checkout master
```

Step 6: Add a file “hello.xml” to the master with different content

```
echo"<greeting>Hello from master branch</greeting>"> hello.xml
git add hello.xml
git commit -m"Add hello.xml in master with different content"
```

Step 7: Observe the log

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

Step 8: Check the differences with Git diff tool

```
git diff master GitWork
```

Step 9: Use P4Merge tool for better visualization

```
git mergetool --tool=p4merge
```

Step 10: Merge the branch to master

```
git merge GitWork
```

Step 11: Observe the git markup

Git will show a conflict in hello.xml with markers like:

```
<<<<<<< HEAD
<greeting>Hello from master branch</greeting>
=======
<greeting>Hello from GitWork branch</greeting>
<note>This is an update in GitWork branch</note>
>>>>>>> GitWork
```

Step 12: Use 3-way merge tool to resolve the conflict git mergetool --tool=p4merge

Manually edit to keep the desired content. For example:

<*greeting*>Hello from both branches</*greeting*>
<*note*>This is an update in GitWork branch</*note*>

Step 13: Commit the changes after conflict resolution

```
git add hello.xml
git commit -m"Resolve merge conflict in hello.xml"
```

Step 14: Observe git status and add backup file to .gitignore

```
git status
echo"*.orig">> .gitignore
git add .gitignore
git commit -m"Add backup files to .gitignore"
```

Step 15: List out all available branches

```
git branch
```

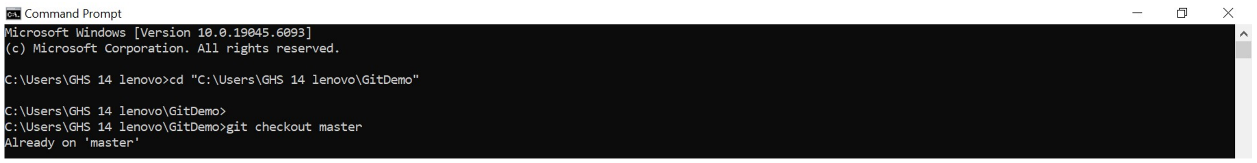
Step 16: Delete the merged branch

```
git branch -d GitWork
```

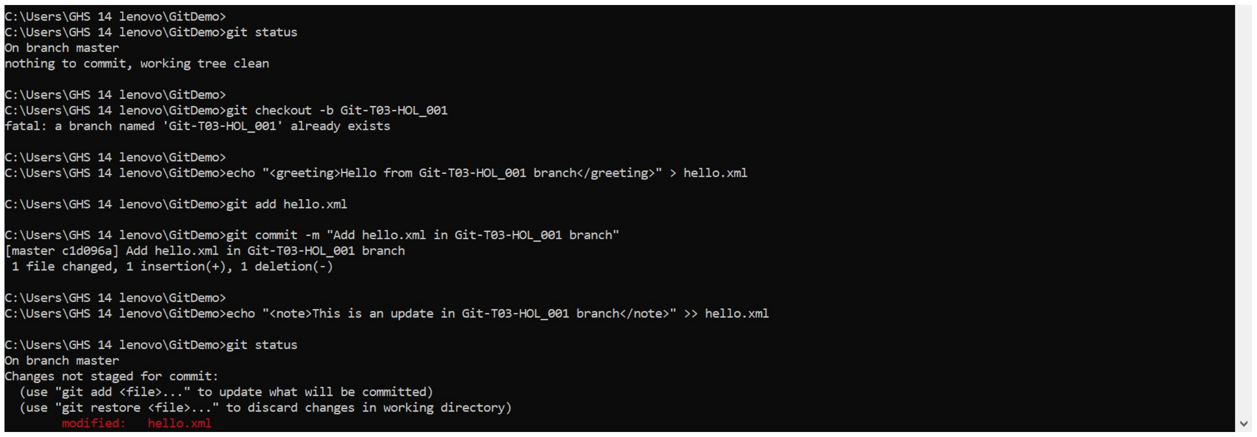
Step 17: Observe the final log

```
git log --oneline--graph--decorate
```

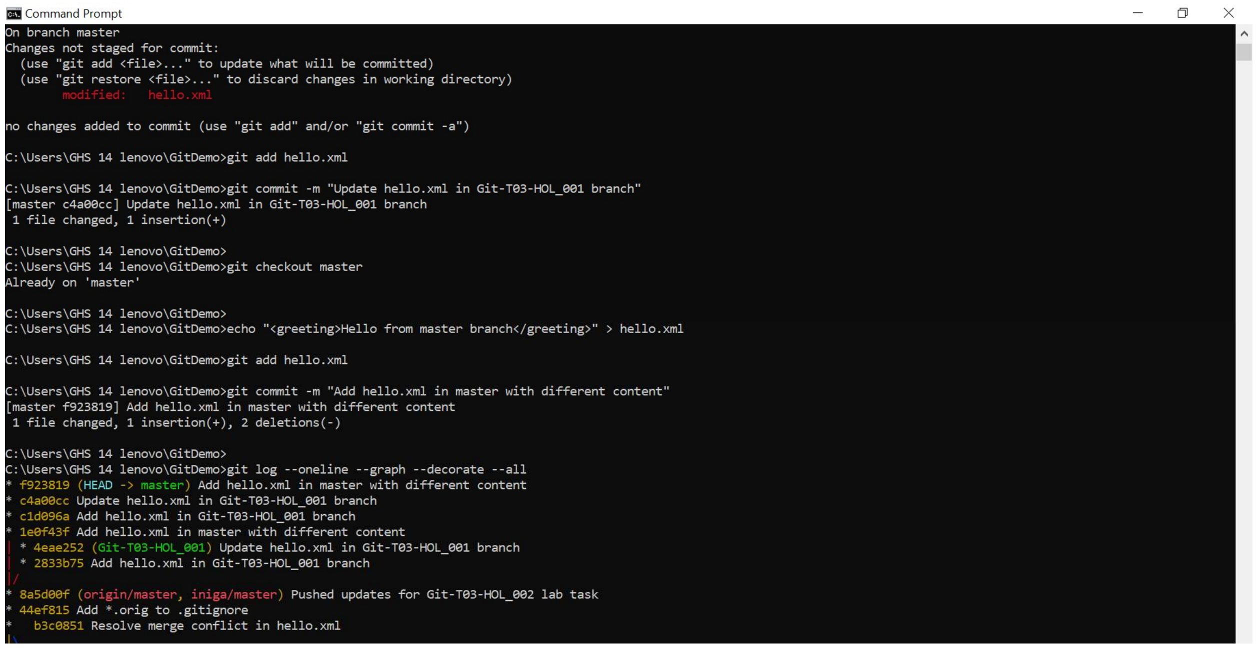
Output:



```
Microsoft Windows [Version 10.0.19045.6093]
(c) Microsoft Corporation. All rights reserved.

C:\Users\GHS 14 lenovo>cd "C:\Users\GHS 14 lenovo\GitDemo"

C:\Users\GHS 14 lenovo\GitDemo>
C:\Users\GHS 14 lenovo\GitDemo>git checkout master
Already on 'master'
```

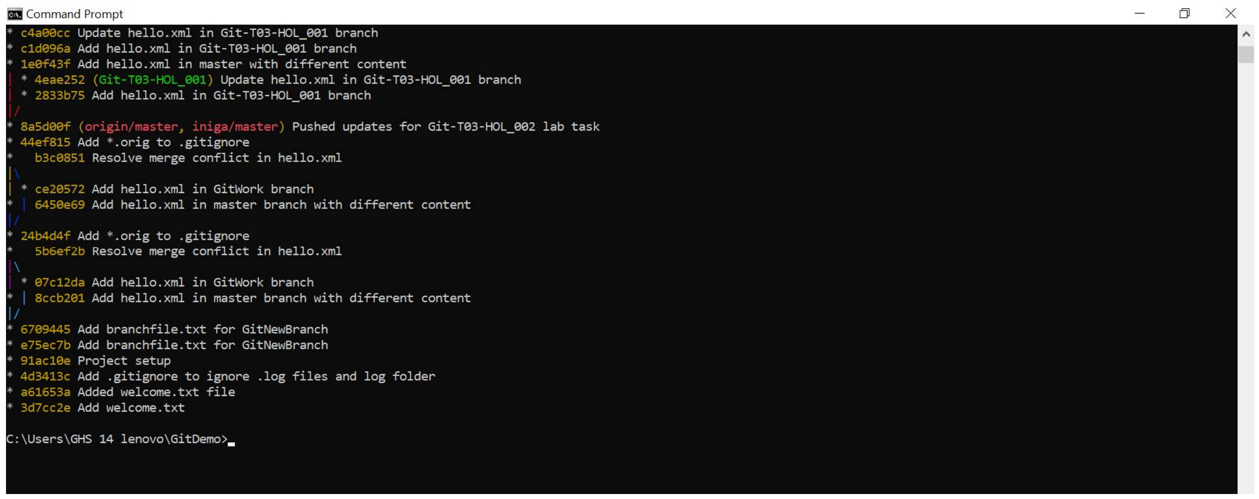


```
C:\Users\GHS 14 lenovo\GitDemo>
C:\Users\GHS 14 lenovo\GitDemo>git status
On branch master
nothing to commit, working tree clean

C:\Users\GHS 14 lenovo\GitDemo>
C:\Users\GHS 14 lenovo\GitDemo>git checkout -b Git-T03-HOL_001
fatal: a branch named 'Git-T03-HOL_001' already exists

C:\Users\GHS 14 lenovo\GitDemo>
C:\Users\GHS 14 lenovo\GitDemo>echo "<greeting>Hello from Git-T03-HOL_001 branch</greeting>" > hello.xml

C:\Users\GHS 14 lenovo\GitDemo>git add hello.xml

C:\Users\GHS 14 lenovo\GitDemo>git commit -m "Add hello.xml in Git-T03-HOL_001 branch"
[master c1d096a] Add hello.xml in Git-T03-HOL_001 branch
 1 file changed, 1 insertion(+), 1 deletion(-)

C:\Users\GHS 14 lenovo\GitDemo>
C:\Users\GHS 14 lenovo\GitDemo>echo "<note>This is an update in Git-T03-HOL_001 branch</note>" >> hello.xml

C:\Users\GHS 14 lenovo\GitDemo>git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   hello.xml
```



Command Prompt

```
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   hello.xml

no changes added to commit (use "git add" and/or "git commit -a")

C:\Users\GHS 14 lenovo\GitDemo>git add hello.xml

C:\Users\GHS 14 lenovo\GitDemo>git commit -m "Update hello.xml in Git-T03-HOL_001 branch"
[master c4a00cc] Update hello.xml in Git-T03-HOL_001 branch
 1 file changed, 1 insertion(+)

C:\Users\GHS 14 lenovo\GitDemo>
C:\Users\GHS 14 lenovo\GitDemo>git checkout master
Already on 'master'

C:\Users\GHS 14 lenovo\GitDemo>
C:\Users\GHS 14 lenovo\GitDemo>echo "<greeting>Hello from master branch</greeting>" > hello.xml

C:\Users\GHS 14 lenovo\GitDemo>git add hello.xml

C:\Users\GHS 14 lenovo\GitDemo>git commit -m "Add hello.xml in master with different content"
[master f923819] Add hello.xml in master with different content
 1 file changed, 1 insertion(+), 2 deletions(-)

C:\Users\GHS 14 lenovo\GitDemo>
C:\Users\GHS 14 lenovo\GitDemo>git log --oneline --graph --decorate --all
* f923819 (HEAD -> master) Add hello.xml in master with different content
* c4a00cc Update hello.xml in Git-T03-HOL_001 branch
* c1d096a Add hello.xml in Git-T03-HOL_001 branch
* 1e0f43f Add hello.xml in master with different content
| * 4eae252 (Git-T03-HOL_001) Update hello.xml in Git-T03-HOL_001 branch
| * 2833b75 Add hello.xml in Git-T03-HOL_001 branch
|/
* 8a5d00f (origin/master, iniga/master) Pushed updates for Git-T03-HOL_002 lab task
* 44ef815 Add *.orig to .gitignore
*   b3c0851 Resolve merge conflict in hello.xml
|\
```



```
* c4a00cc Update hello.xml in Git-T03-HOL_001 branch
* c1d096a Add hello.xml in Git-T03-HOL_001 branch
* 1e0f43f Add hello.xml in master with different content
| * 4eae252 (Git-T03-HOL_001) Update hello.xml in Git-T03-HOL_001 branch
| * 2833b75 Add hello.xml in Git-T03-HOL_001 branch
|/
* 8a5d00f (origin/master, iniga/master) Pushed updates for Git-T03-HOL_002 lab task
* 44ef815 Add *.orig to .gitignore
*   b3c0851 Resolve merge conflict in hello.xml
|\
| * ce20572 Add hello.xml in GitWork branch
* | 6450e69 Add hello.xml in master branch with different content
|/
* 24b4d4f Add *.orig to .gitignore
*   5b6ef2b Resolve merge conflict in hello.xml
|\
| * 07c12da Add hello.xml in GitWork branch
* | 8ccb201 Add hello.xml in master branch with different content
|/
* 6709445 Add branchfile.txt for GitNewBranch
* e75ec7b Add branchfile.txt for GitNewBranch
* 91ac10e Project setup
* 4d3413c Add .gitignore to ignore .log files and log folder
* a61653a Added welcome.txt file
* 3d7cc2e Add welcome.txt

C:\Users\GHS 14 lenovo\GitDemo>
```

5.GIT-HOL

GIT CleanUp and PushBack

After completing your local changes in Git, it's essential to clean up your working environment and push those changes to the remote repository. This ensures that your team members have access to the latest code and the repository remains in sync. In this lab, you will go through the necessary Git commands to clean up your local repository and successfully push your final changes to GitHub.

Objectives:

1. Understand how to verify a clean working state before pushing.
2. Learn how to pull the latest changes from the remote repository.
3. Execute Git commands to push local changes to the remote repository.
4. Observe and confirm the successful reflection of changes in the remote GitHub repository.

Instructions & Commands

1. Verify if master is in clean state

```
git status
```

2. List out all the available branches

```
git branch -a
```

3. Pull the remote git repository to the master

```
git checkout master
git pull origin master
```

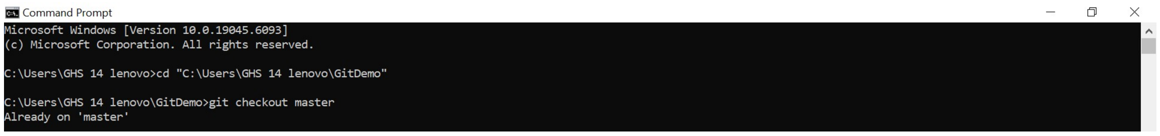
4. Push the changes from "Git-T03-HOL_002" branch to remote

```
git checkout Git-T03-HOL_002
git add .
git commit -m "Pushed updates for Git-T03-HOL_002 lab task"
git push origin Git-T03-HOL_002
```

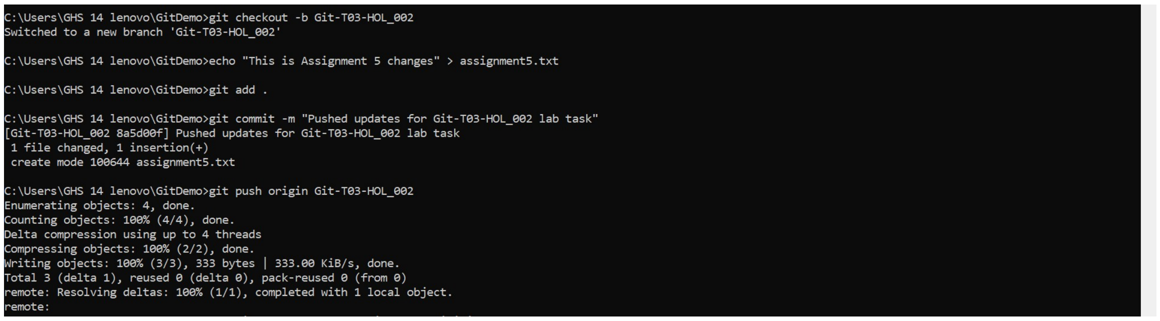
5. Observe if the changes are reflected in the remote repository

    - Go to your GitHub repository in the browser.
    - Navigate to the branch Git-T03-HOL_002.
    - Confirm that the files/changes are visible.

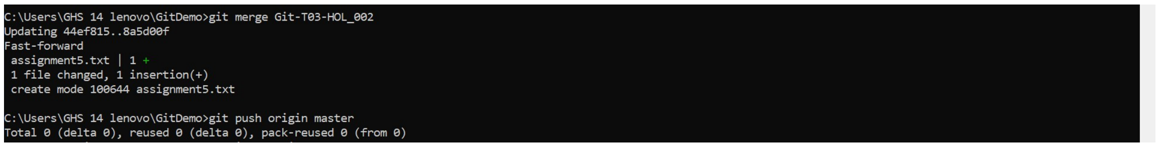
Output:



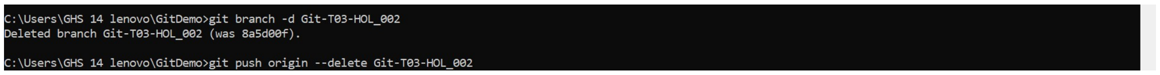
```
Command Prompt
Microsoft Windows [Version 10.0.19045.6093]
(c) Microsoft Corporation. All rights reserved.

C:\Users\GHS 14 lenovo>cd "C:\Users\GHS 14 lenovo\GitDemo"

C:\Users\GHS 14 lenovo\GitDemo>git checkout master
Already on 'master'
```



```
C:\Users\GHS 14 lenovo\GitDemo>git checkout -b Git-T03-HOL_002
Switched to a new branch 'Git-T03-HOL_002'

C:\Users\GHS 14 lenovo\GitDemo>echo "This is Assignment 5 changes" > assignment5.txt

C:\Users\GHS 14 lenovo\GitDemo>git add .

C:\Users\GHS 14 lenovo\GitDemo>git commit -m "Pushed updates for Git-T03-HOL_002 lab task"
[Git-T03-HOL_002 8a5d00f] Pushed updates for Git-T03-HOL_002 lab task
 1 file changed, 1 insertion(+)
 create mode 100644 assignment5.txt

C:\Users\GHS 14 lenovo\GitDemo>git push origin Git-T03-HOL_002
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 333 bytes | 333.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
```



```
C:\Users\GHS 14 lenovo\GitDemo>git merge Git-T03-HOL_002
Updating 44ef815..8a5d00f
Fast-forward
 assignment5.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 assignment5.txt

C:\Users\GHS 14 lenovo\GitDemo>git push origin master
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
```



```
C:\Users\GHS 14 lenovo\GitDemo>git branch -d Git-T03-HOL_002
Deleted branch Git-T03-HOL_002 (was 8a5d00f).

C:\Users\GHS 14 lenovo\GitDemo>git push origin --delete Git-T03-HOL_002
```